有限元理论基础及Abaqus内部实现方式研究系列21： 自主CAE开发实战经验第二阶段总结

Theoretical Foundation of Finite Element Method and Internal Implementation of Abaqus: Series 21 - Summary of the Second Stage of Autonomous CAE Development Experience

SnowWave02 关注 Focus

2020年4月27日 14:25 April 27, 2020 14:25 浏览：3103 Views: 3103 评论：5 Comments: 5

（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21： 自主CAE开发实战经验第二阶段总结的图1

有限元理论基础及Abaqus内部实现方式研究系列21： 自主CAE开发实战经验第二阶段总结的图2

==概述== ==Overview==

有限元理论基础及Abaqus内部实现方式研究系列21： 自主CAE开发实战经验第二阶段总结的图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.

MSC Software
HFSS
FEKO
ANSYS
MSC Nastran
Altair HyperWorks
Optimus
ABAQUS
LS-DYNA
FLUENT
Tecplot 10
ModelCenter
$\dot{\mathbf{u}} = \sum_b N_b(\mathbf{x})\dot{\mathbf{u}}_b(t) = \mathbf{N}(\mathbf{x})\dot{\mathbf{u}}(t)$
$\Pi(\mathbf{U}) = \int_\Omega W(C_{IJ})\,d\Omega - \Pi_{ext}$

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21篇：自主CAE开发实战经验第二阶段总结==

==21st Article: Summary of the Second Stage of Autonomous CAE Development Experience==

有限元基础理论和Abaqus内部实现方式研究系列的文章已经完成了20篇，我们每十篇将写一篇自主CAE的总结，网上分析自主CAE的得失的文章已经很多了，我们站的角度不同，不是从国家战略的层面上来呼吁自主CAE的重要性，这边的总结仅仅结合结构有限元求解器iSolver的开发具体谈谈实现自主CAE过程中的切身经验和体会。上一次的总结文章有兴趣的可以看下面的链接：

The series of articles on the fundamental theory of finite element analysis and the internal implementation of Abaqus has been completed with 20 articles. We write a summary of independent CAE every ten articles. There have been many articles analyzing the pros and cons of independent CAE online, but our perspective is different; we are not calling for the importance of independent CAE from the level of national strategy. This summary simply discusses the personal experiences and insights gained from the process of realizing independent CAE, specifically focusing on the development of the structural finite element solver iSolver. Those interested in the previous summary can view the link below:

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

本文主要是第二阶段的开发经验总结，iSolver上一阶段主要是技术功能的开发研究，而第二阶段在开发新功能的同时，在推广和工程化应用的过程中有了更深的体会。

This article mainly summarizes the development experiences of the second phase. In the first phase, iSolver mainly focused on the research and development of technical functions, while in the second phase, deeper insights were gained in the process of promoting and engineering application of new functions.

1.1 十年实现自主CAE很难，那要是一百年呢?

1.1 It is difficult to achieve independent CAE in ten years, what if it takes a hundred years?

自主CAE软件，无论是理论水平还是软件开发能力，国内比我们做的好的比比皆是，有些不愿意公开宣传，有些在满头练内功。我们更希望和大家交流，得到大家的反馈，深入的讨论具体怎么实现自主CAE的方案和实践。也许我们这一代永远替换不了商软CAE，但至少我们可以一起策划未来怎么替代，替代过程中有哪些雷区我们可以先尝试一下，节约后来者的研究时间，并且可以留下个样板房，大家可以从中看到自主CAE不是那么遥不可及，还可以像愚公移山一样留给下一代人去解决。iSolver的目标也是希望做一个求解器的开发平台，大家可以在这个平台上更加快捷和深入的学习有限元知识，对国内做自主CAE的人也可以降低编程难度，当效率提高了，才可能省下时间来专心研究算法和后面的商业化推广，很多时候开发自主CAE的同行并不缺少技术和推广的能力，也能找到些自由支配的时间，但可惜很多时候被开发效率和协同而白白浪费了，而且很难积累。自主CAE需要的是百花齐放百家争鸣，现在用户对国内的CAE基本不相信，无论你是做什么行业什么专业的，当大家都花时间在潜心研究自主CAE

时，才可能把自主CAE真正做到实处，给客户信心，新的自主CAE也更容易被客户接受。

Independent CAE software, whether in terms of theoretical level or software development capabilities, there are many in our country that are better than us, some are unwilling to publicize, and some are practicing their internal skills. We hope to communicate with everyone and get everyone's feedback, and have in-depth discussions on how to implement independent CAE solutions and practices. Perhaps our generation will never replace commercial CAE software, but at least we can plan together on how to replace it, what pitfalls we can try first in the process of replacement, save the research time of later generations, and leave a model house for everyone to see that independent CAE is not so distant. It can also be left for the next generation to solve like Yu Gong moving the mountain. The goal of iSolver is also to build a solver development platform, where everyone can learn finite element knowledge more quickly and deeply, and it can also reduce the programming difficulty for those who do independent CAE in our country. When efficiency is improved, it is possible to save time to focus on algorithm research and subsequent commercial promotion. Many times, those who develop independent CAE are not short of technical and promotional capabilities, and they can also find some free time, but unfortunately, it is often wasted on development efficiency and collaboration, and it is difficult to accumulate. What independent CAE needs is a flourishing of various schools of thought. Now, users basically do not believe in domestic CAE, no matter what industry or professional field you are in. Only when everyone spends time on in-depth research of independent CAE can it be truly implemented, give customers confidence, and make new independent CAE easier for customers to accept.

现在的国外CAE一套动辄数百万，这个定价明显虚高，小企业根本承担不起，很大程度是垄断造成的，只有中国国内有了可替代的方案，国外的CAE软件的价格才会主动降下来，这对我们每个企业来说可以省掉一大笔经费。而且国外CAE还有各种限制措施，国防关键工程不能用，license节点数目受限，内部计算方式不公开，这些都需要国内自主CAE成长起来才有谈判资格。

Now, foreign CAE software is priced at several million at a time, which is obviously overpriced, and small and medium-sized enterprises can't afford it. This is largely due to monopoly. Only when China has a viable alternative, will the price of foreign CAE software actively drop, which can save a lot of funds for each of us. Moreover, foreign CAE has various restrictions, such as not being able to be used in national defense key projects, limited license node numbers, and non-public internal calculation methods. These require the growth of domestic independent CAE to have the qualifications for negotiation.

1.2 拨开重重的迷雾，商业CAE软件开发还是有迹可循的

1.2. Unveiling the mysteries, there are still traces to be followed in the development of commercial CAE software.

1.2.1 商业化的思维 1.2.1. Commercial thinking.

很多做技术的人都看不起商业化，那是不是可能因为恰恰是我们商业化做的不好或者根本无力做商业化推广呢？我们以前一直从事技术方面的开发工作，以前觉得技术是最重要的，但在自主CAE商业化推广摸索一段时间后，才发现技术是最次要的，商业化推广更重要，当你商业化做的好的话技术是可以用商业搞定的，看看最近几年大厂商在不断的合并就知道了。

Many people in the technology field look down upon commercialization, could it be because we just don't do commercialization well or are completely unable to promote it? We have always been engaged in technical development, and previously thought that technology was the most important. However, after some exploration in the promotion of independent CAE commercialization, we found that technology is actually the least important, and commercial promotion is more crucial. When you do commercialization well, technology can be resolved through commercial means. Just look at the continuous mergers of large manufacturers in recent years.

在以前，商业化肯定只能是靠金钱才能堆出来，但现在网络资源很多，传播途径和范围也可能现象级的，同时有些成本都可以忽略，这也给了大家更多的机会方便的宣传自己的产品，使得自主CAE的宣传有可能和商软在同一个舞台了，就像现在的技术邻一样。

In the past, commercialization could only be achieved through money, but now with abundant online resources and the possibility of exponential spread of dissemination channels, some costs can even be ignored. This has given everyone more opportunities and convenience to promote their products, making it possible for independent CAE promotion to be on the same stage as commercial software, just like the current "Technical Neighbor" platform.

shutterstock.com • 1006041130

1.2.2 学会取舍 1.2.2 Learn to make trade-offs

几乎所有的工程师都会梦想提出一套方案解决所有问题，并为之付诸半生的努力奋斗，最终毫无意外地折戟沉沙。需求是无限的，软件能力是有限的，明确的定位和务实的发展计划是自主CAE软件一开始就需要考虑的。国内许多高校和研究机构曾经研发出了优秀的CAE软件，却没有能够坚持下来，除了受国外商软的冲击外，很大一部分源于自身的定位迷失和不合理的发展计划，在某些项目中成功实施后，胆大的全线铺开大踏步前进，死在追求“通用”的路上；胆小的犹犹豫豫原地徘徊，迷失在寻找“通用”的路上。“通用”似乎是一个有魔力的词，是所有软件都绕不开的坑。直至目前为止，国内的大多数研究机构和高校仍然沉迷于通用数据格式、通用数据接口、通用功能等

字眼，却不知若干年前有一种词叫“中性”。取舍是这个阶段最重要的能力和品质，对需求进行取舍，在取舍的过程中把握住行业的动向，看清楚自身的能力和特点，明确自身定位，制定切实可行的发展计划。

Almost all engineers dream of proposing a solution to solve all problems and dedicating half of their lives to struggling for it, only to end up in failure without any surprise. Needs are infinite, while software capabilities are limited. Clear positioning and practical development plans are what autonomous CAE software needs to consider from the beginning. Many universities and research institutions in China have developed excellent CAE software, but they have not been able to sustain it. In addition to the impact of foreign commercial software, a large part of the reason is the loss of self-positioning and unreasonable development plans. After successfully implementing certain projects, some people boldly expand all-out and advance in leaps and bounds, only to die on the road to pursuing "universality"; others hesitate and 徘徊, lost in the search for "universality". "Universality" seems to be a magical word, a pit that all software cannot avoid. Until now, most research institutions and universities in China are still 沉迷于 the concepts of universal data formats, universal data interfaces, and universal functions, but they do not know that there was a word called "neutral" years ago. Choice-making is the most important ability and quality at this stage. It is necessary to make choices for the needs, grasp the trends of the industry in the process of choice-making, understand one's own capabilities and characteristics, clarify one's own positioning, and formulate practical development plans.

1.2.3 应用为王 1.2.3 Application is King

应用有两种： Two applications are used:

第一种是深度：做个超大工程证明自主CAE的求解能力，千万网格量级以上，多少p以上级别的超算资源，而且还要连续算个几天几夜以上，这种高档饭店的应用一个完整的例子就能测出程序的计算速度和内存，优化你的程序结构。优点是展示度好，领导们重视，缺点是一个项目做完，很难推广到另一个项目，再重来一遍，从另一方面来

说，这也是优点，一次做完了，后面哪还有项目。

The first is depth: to do a super large project to prove the computational ability of independent CAE, with a grid quantity of tens of millions or even more, and supercomputing resources of p-level or above, and it needs to be calculated continuously for several days and nights. This kind of high-end restaurant application can measure the calculation speed and memory of the program with a complete example, and optimize the program structure. The advantage is that it has a good display, and leaders pay attention to it. The disadvantage is that after a project is completed, it is difficult to promote it to another project, and it needs to be done all over again. From another perspective, this is also an advantage, as there are no projects left after it is done.

第二种是广度：就是大众化的谁都可以拿来直接算算的小问题，这种问题一般几万个网格就够了，普通pc机上都可以运行，时间差不多5分钟也就运行完了。如果你的目标是产品开发，那么对程序开发起到更大作用的是这种路边大排档的应用，这种应用才能给自主CAE快速的反馈，形成批量的改进意见，指引我们产品的开发方向，然后产品版本不断的迭代。

The second is breadth: it is a common problem that anyone can use directly to calculate, and such problems generally require only tens of thousands of grids, and can be run on a regular PC, and it takes about 5 minutes to complete. If your goal is product development, then the application of this kind of roadside snack bar plays a greater role in program development. This kind of application can provide rapid feedback for independent CAE, form a batch of improvement suggestions, and guide the development direction of our products, and then the product version is continuously iterated.

现在国内自主CAE基本都是第一种大工程的应用，第二种普通的应用难做、没展示效果，而且也没法考核。所以现在的课题等基本都是以XX大型项目为示范验证，导致很多做自主CAE为了完成项目更是只能花大把时间去做前面点上的应用，而没有精力去做后者面上的应用。

Currently, domestically developed CAE is mostly applied in large-scale projects, and it is difficult to implement and demonstrate the effects of the second type of general application, and it is also not possible to evaluate it. Therefore, the current topics are basically verified with XX large projects as examples, leading to many who develop independent CAE having to spend a lot of time on the applications marked in the early stage of the project, and not having the energy to do the applications on the latter surface.

第二种普通的应用难做，我们觉得相对技术来说，另两个问题更突出：

The second type of common application is difficult to do, and we think that the other two issues are more prominent compared to the technology

（1） 用户对商软的深度依赖。也许有一天国外断供，打开软件发现无法启动然后又要完成以前的任务时，可能才能意识到自己对商软是何等的依赖。

(1) The deep dependence on commercial software. Perhaps one day, when the supply from abroad is cut off and the software cannot be started, and when you have to complete previous tasks, you may realize how much you depend on commercial software.

（2） 用户对国内自主CAE软件的不信任。不仅是结果的正确性，还有软件的用户体验和发展前景等，转换平台是要花费成本的，不光是时间，还有对未来的投资。

(2) The distrust of domestic independent CAE software among users. This includes not only the correctness of the results but also the user experience and development prospects of the software. Shifting platforms involves costs, not just time but also investment in the future.

而现在应用国外商软占垄断地位的情况下，如何在普通的应用上推广自主CAE呢？我们觉得必须和商软结合，接口按商软标准开发，和商软对接转换数据，甚至在商软平台上做二次开发和集成，做那些只是修修补补、搞个界面看起来毫无技术含量的定制，在商软的世界中摸爬滚打，一个项目也许你99%的工作都献给了商软，但至少还有1%的经验留给了自己，那么等你做到100个仿真二次开发和商软集成的项目时，你也可以第一线理解商软的应用场景，理解客户的业务流程，从而基本了解自主CAE往后怎么应用了。这也是商软的核心，只是大部分人都认为二次开发就是搞个界面最没有技术含量，领导不重视，也没经费，更不能发文章。黑猫白猫，抓到老鼠就是好猫，对客户来说，只要能满足我的业务需求就行，至于底下是不是商软还是自主的，不重要，等到了你能编出上层的业务界面后，那么底层才有机会用自主CAE来替代商软。

Now that foreign commercial software occupies a dominant position, how can we promote independent CAE in common applications? We believe that it must be integrated with commercial software, with interfaces developed according to commercial software standards, and data conversion with commercial software. Even on the commercial software platform, we can do secondary development and integration, such as customizing those that only require minor repairs and a superficial interface with no technical value. In the world of commercial software, you may have dedicated 99% of your work to it, but at least 1% of the experience is left for yourself. By the time you have completed 100 simulation secondary development and integration projects with commercial software, you can also understand the application scenarios of commercial software on the front line, understand the business processes of customers, and thus basically understand how independent CAE can be applied in the future. This is the core of commercial software, but most people think that secondary development is the least technically valuable, as it is not valued by leaders, lacks funding, and cannot be published as an article. Black cat, white cat, as long as it catches the mouse, it's a good cat. For customers, as long as it meets my business needs, it doesn't matter whether it's commercial software or independent software. Only when you can develop the upper-level business interface will there be an opportunity to replace commercial software with independent CAE.

1.2.4 不缺技术，但缺平台 1.2.4 Not lacking in technology, but lacking in a platform.

国内做有限元理论和算法研究出色的不在少数，在有限元的技术方面国内并不缺少，况且有限元软件总是滞后于最新的有限元理论技术的。但技术好和将技术转化为代码形成产品是两回事，可能是两批人，两个公司，两个领域。整合这些有限元技术，让软件在不断吸收外部技术的同时还能往前发展，需要的是一个技术积累的平台。这个平台

我们认为应该具备两个最基本的特点：

There are not a few people in China who excel in finite element theory and algorithm research, and there is no lack of technology in finite element in China. Moreover, finite element software always lags behind the latest finite element theory and technology. However, being good at technology and converting technology into code to form a product is a different matter, which may involve two groups of people, two companies, and two fields. To integrate these finite element technologies, allowing the software to develop while continuously absorbing external technologies, what is needed is a platform for technical accumulation. We believe that this platform should have two basic characteristics:

（1） **可扩展。**这个平台首先要求是架构完善，可扩展性很强，已经考虑了对未来的技术方面的积累，新功能在平台中能找到位置。这个需要花费大量的时间去调研，甚至在你编第一行代码前，如果你做的CAE软件有对标软件的话，那么就对标商软的架构研究是一个好途径，商软是个黑盒子，但很多情况在外面的表现也可以猜到内部是怎么实现的，然后再找开源代码或者编写小程序去证明一下。同时开源代码的架构也是不错的研究对象（如果是研究开源的算法，还是要谨慎的，商软一般都会修正）。

(1) Scalable. The first requirement for this platform is a complete architecture with strong scalability, which has already considered the accumulation of future technological aspects, and new functions can be found in the platform. This requires a lot of time for research, and even before you write the first line of code, if your CAE software has a comparable software, studying the architecture of the commercial software is a good approach. The commercial software is a black box, but in many cases, its external performance can also be guessed at the internal implementation. Then, find open-source code or write small programs to prove it. At the same time, the architecture of open-source code is also a good research object (if you are studying open-source algorithms, you still need to be cautious, as commercial software usually makes corrections).

（2） **易维护。**这个平台的快速迭代和维护应该也是在你的团队能力之内的。在国内，除开个别国家单位，自主CAE在实现商业化应用前都是纯投入，而且没有个几年时间基本不会出成果，上面的领导和一起作战的兄弟们都需要养家糊口完成经济指标，只能挤时间开发，那么平台的可维护性就极其重要。在外人看来，只要技术上可行，时间不是问题，可以加人、可以加班。要是协同和加班能搞定的都是小问题，平台有些技术上的缺陷很可能是人数越多开发越慢，团队所有人7X24小时还完不成的。现在很多都基于开源代码开发，这个要分情况的，如果你不准备修改开源代码，那么没问题，集成个初步版本看到效果是很快的，后面开源软件升级时还能一块升级。但如果你后面还需要维护和功能的添加，那就很考验团队消化能力了，不是自己编的代码理解是要花大量时间的，在整个可能超过10年的CAE开发过程中，基于开源代码花费的时间不一定比从零开发少，而且，还没有知识产权，有时还不

如自己开发，就当是一场马拉松，不在乎起跑，只在乎终点。

(2) Easy to maintain. The rapid iteration and maintenance of this platform should also be within the capabilities of your team. In China, except for a few national units, independent CAE is purely an investment before achieving commercial application, and it usually takes several years to produce results. The leaders and brothers fighting together need to support their families and complete economic indicators, so they can only squeeze time for development. Therefore, the maintainability of the platform is extremely important. From the outside, as long as the technology is feasible, time is not an issue, you can add people, you can work overtime. If collaboration and overtime can solve the problem, it is a minor issue. Some technical defects of the platform may become slower to develop as the number of people increases, and the entire team has to work 7x24 hours to complete it. Now many are based on open-source code development, and this needs to be handled on a case-by-case basis. If you do not plan to modify the open-source code, then there is no problem, integrating an initial version to see the effect is very quick, and it can be upgraded together with the upgrade of the open-source software. But if you need to maintain and add functions later, it will test the team's digestion ability, and it takes a lot of time to understand the code that is not written by yourself. Throughout the entire CAE development process that may exceed 10 years, the time spent on open-source code may not be less than that spent on starting from scratch, and there is no intellectual property rights, and it may not be as good as self-development. It is like a marathon, not caring about the starting point, but only about the finish line.

平台的优劣直接决定了积木垒起来的高度，决定你的技术转换为软件代码所能达到的高度。平台没做好，现有的代码就如同建立在流沙上的房子一般，总有一天会被冲垮，甚至是现有代码越多，反而为了兼容以前的代码更是难以发展，只能带着脚链前行，前面的坑还没填好，后面又在坑上建房子了。

The strengths and weaknesses of the platform directly determine the height of the building blocks stacked together, determining the height your technology can reach when converted into software code. If the platform is not well done, the existing code is like a house built on quicksand, it will be washed away one day, and even more code may make it even harder to develop due to the need to be compatible with previous code, and you can only move forward with shackles. The holes in front have not been filled, and houses are being built on the holes in the back.

1.3 前面依然是商软CAE的崇山峻岭，但至少我们找到了一个上山的平台

1.3 The path ahead is still steep with commercial CAE software, but at least we have found a platform to climb up.

在研究商软实现方式的过程中，很多次发现商软都比我们预想的复杂，譬如壳单元的刚度矩阵，我们以为研究清楚了Abaqus中S4R、Nastran的Quad4等的修正方式，然后编程在iSolver中实现，结果却发现商软后面有更多的修正，再譬如虽然我们现在做的是求解器，但我们知道后面还有“前后处理”更麻烦，远比我们做求解器花的时间更多。商软后台的有限元大师和软件架构工程师们就像一个个绝世的高手，编写了绝妙的解决工程问题的算法，在一个我们现在都无法测量高度的山峰上等待着别人去探寻，同时设置了无数的小山坡消耗你的体力。高手们可以腾云驾雾，但我等资质平庸之辈只能老老实实一个一个山坡努力的往上爬，希望有朝一日领略到最后的山峰有多高，多雄伟。

During the research of commercial software implementation methods, many times we found that commercial software is more complex than we expected. For example, the stiffness matrix of shell elements, we thought we had understood the correction methods of S4R in Abaqus and Quad4 in Nastran, and then implemented them in iSolver, only to find that there are more corrections in the commercial software. For example, although we are currently doing solvers, we know that there are more complicated "pre-processing" and "post-processing" ahead, which take much more time than we spend on solvers. The finite element masters and software architects behind the commercial software are like a group of world-class experts, who have written excellent algorithms to solve engineering problems, waiting for others to explore on a mountain peak that we cannot measure its height, while setting countless small hills to consume your energy. Experts can soar, but we, the ordinary people, can only climb one hill at a time, hoping to one day understand how high and magnificent the final peak is.

我们并不是什么理论大牛，可能理论水平还没有读文章的你高，这么多年来一直在发展进步仅仅是我们有一个还算完善的有限元开发平台iSolver。这个平台还有很多不足之处，但起码满足了前面我们所说的两个特点：

We are not theoretical experts, and our theoretical level may not be as high as yours. Over the years, we have been making progress simply because we have a relatively complete finite element development platform, iSolver. This platform still has many shortcomings, but it at least meets the two characteristics we mentioned earlier:

（1） **可扩展。**整体按增量迭代法的流程实现，功能点的扩展都以子程序的形式单独添加，子程序的接口按Abaqus标准开发，框架和子程序完全独立，可由不同的人维护。

(1) Extensible. The overall implementation is based on the incremental iterative method, and the expansion of functional points is added in the form of sub-programs separately. The interfaces of the sub-programs are developed according to the Abaqus standard, and the framework and sub-programs are completely independent, which can be maintained by different people.

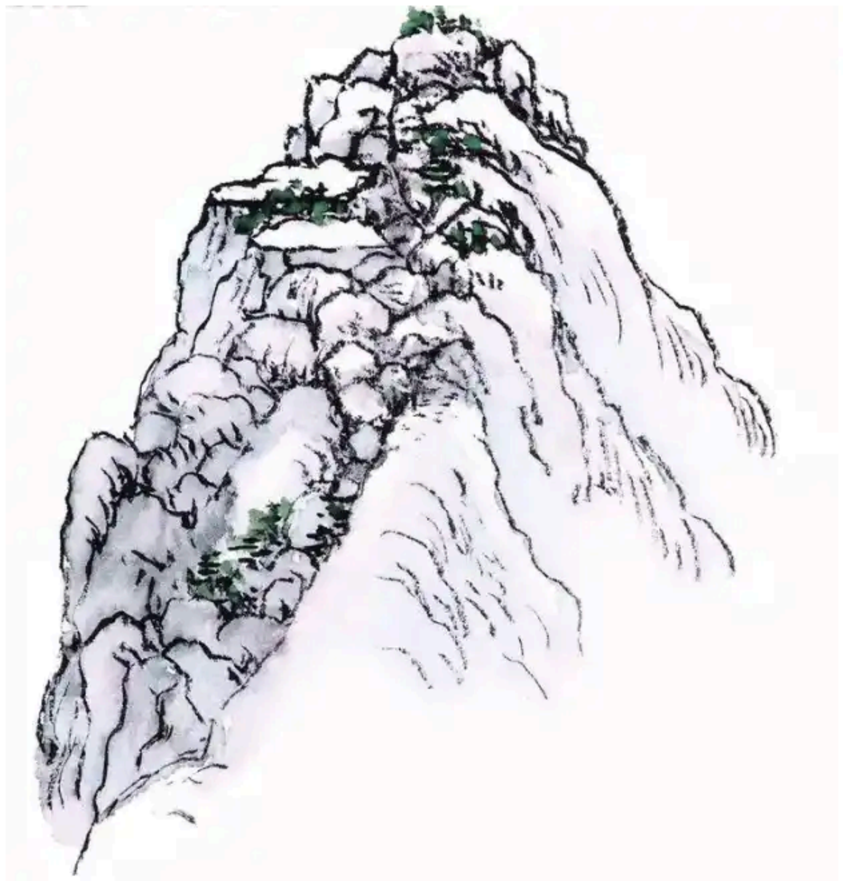
（2） **易维护。**我们从零开始整套代码自己编程实现，对内部的技术细节都非常清楚，新功能点开发和老功能的更改可以在相对较短时间内完成。

(2) Easy to maintain. We have implemented the entire code from scratch ourselves, and we are very familiar with the internal technical details. The development of new functional points and the modification of old functions can be completed in a relatively short period of time.

因此，我们可以在这个平台上不断编程研究有限元理论和商软的算法修正，也能将这种研究后的代码转换为程序放到这个平台中，功能不断的积累，后面研究新理论时可以站在以前的理论基础上前进，形成良性循环。同时，有些

我们实现不了的部分也可以请专业性更强的单位协作开发。

Therefore, we can continuously program and research finite element theory and algorithm corrections of commercial software on this platform, and can also convert the code after this research into a program and place it on this platform, accumulating functions continuously. When studying new theories later, we can move forward on the basis of previous theories, forming a virtuous cycle. At the same time, some parts that we cannot implement can also be developed in collaboration with units with stronger professionalism.



1.4 邀请你一起上山 1.4 Invite you to climb the mountain together

iSolver不仅是个求解器，同时也是个学习有限元和深入了解商软内部实现的平台。光看理论书很多公式都不够直观，而光操作商软无论你花多少时间，总有个天花板在那，只有真正的自己编程才可能将书本理论、商软操作和内部实现三者融会贯通。而iSolver相对商软的二次开发编程来说，学习效率和成本也会降低很多，可以让学习者更多的关注算法本身，而不用花费太多无意义的时间在调试代码上。我们在往后开发iSolver的过程中，也依然会在这个平台上一步步的尝试，如果有了新的功能，也会像现在一样开发相应的接口，配合视频和文章，让后来者在这

个平台上深入理解有限元的基础理论和商软的内部实现。

iSolver is not only a solver but also a platform for learning finite element analysis and gaining in-depth understanding of the internal implementation of commercial software. Just reading theoretical books, many formulas are not intuitive, and operating commercial software alone, no matter how much time you spend, there is always a ceiling. Only by truly programming can the book theory, commercial software operation, and internal implementation be integrated. Compared to the secondary development programming of commercial software, iSolver can also greatly reduce the learning efficiency and cost, allowing learners to focus more on the algorithm itself and not spend too much time on debugging code. In the process of developing iSolver in the future, we will also continue to try step by step on this platform. If there are new features, we will also develop corresponding interfaces as now, in conjunction with videos and articles, so that latercomers can deeply understand the basic theories of finite element analysis and the internal implementation of commercial software on this platform.

如果有兴趣，你可以现在就下载iSolver尝试一下，照着我们视频中的例子做做看，纸上得来终觉浅，须知此事要躬行。在学习的同时你也可以开发属于自己的材料、单元子程序，并形成自己的算法库，在iSolver平台上不断积累，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。

If you are interested, you can download iSolver now and try it out. Follow the examples in our video and see for yourself. Knowledge gained from books is shallow, and one must practice to truly understand. While learning, you can also develop your own materials, element subroutines, and form your own algorithm library, continuously accumulating on the iSolver platform. The interfaces of iSolver subroutines are fully implemented according to Abaqus standards, and the Abaqus subroutine interfaces have not changed much in recent years. Similarly, although iSolver is constantly evolving, the iSolver subroutine interfaces will remain unchanged. All algorithm subroutines written on iSolver only need to maintain their own algorithm part, rather than the entire finite element solution process.

也希望你像我们一样有一天突然觉得，原来自主的CAE也不是难于上青天，或许我也可以编程试试，一旦你开始自己编程，那么恭喜你，你将进入另一个世界。也祝愿你有一天也不需要使用iSolver，而是在自己的平台上摸索发展，国内的CAE的发展需要更多的自主平台。

I also hope that one day, like us, you suddenly realize that autonomous CAE is not as difficult as climbing to the sky. Perhaps I can try programming as well. Once you start programming, congratulations, you will enter another world. I also wish that one day you will not need to use iSolver, but explore and develop on your own platform. The development of domestic CAE needs more autonomous platforms.

（1）自主结构有限元开发框架iSolver下载地址：

(1) Download address of the autonomous structural finite element development framework iSolver:

http://www.jishulink.com/content/post/337351

1) 1Mb的Abaqus插件程序 1) 1Mb Abaqus plugin program

2) 1分钟的安装过程 2) 1-minute installation process

3) 和Abaqus完全一致的使用方式

3) Exactly the same usage as Abaqus

4) 编程效率远超Fortran的Matlab子程序开发

4) Programming efficiency far exceeds Matlab subroutine development in Fortran

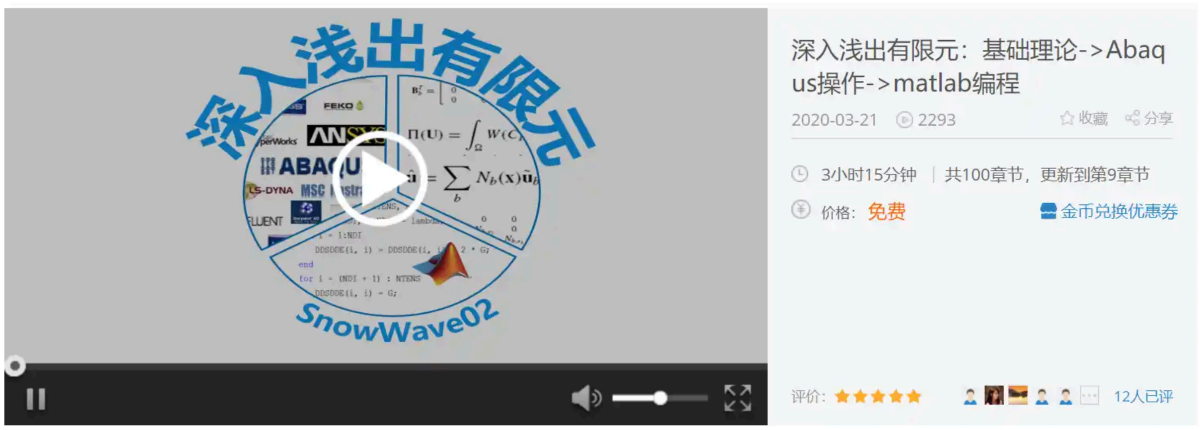
（2）邀请你一起学习有限元基础理论和编写单元程序：

(2) Invite you to learn the basic theory of finite element analysis and write element programs:

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 2293 收藏 分享

3小时15分钟 | 共100章节，更新到第9章节

价格：免费 金币兑换优惠券

评价：★★★★★ 12人已评

（3）邀请你一起学习有限元材料理论和编写材料程序：

(3) Invite you to learn finite element material theory and write material programs together:

https://www.jishulink.com/college/video/c13034

Abaqus用户子程序UMat详解与开发工具 Abaqus User Subroutine UMat Explanation and Development Tool

Abaqus用户子程序UMat详解与开发工具（未完，待续）

2019-01-21 10883 收藏 分享

5小时10分钟 | 共100章节，更新到第14章节

价格：￥10.00 金币兑换优惠券

VIP特价：￥9.00

评价：★★★★★ 26人已评

1.5 特别感谢 1.5 Special Thanks

（1）　特别感谢技术邻，给了我们一个自由发挥、展示的平台

(1) Special thanks to Technical Neighbor, which gave us a platform to freely express and showcase ourselves.

（2）　感谢开发过程中给过我们指导、帮忙一起校核公式、调试程序结果、提供参考代码的同行。

(2) Thanks to the colleagues who provided guidance, helped to verify formulas, debug program results, and provided reference code during the development process.

（3）　感谢线上线下交流、帮我们找问题、给我们反馈意见、指引我们方向的iSolver用户，也许我们对你没有多少作用，反而是你的意见对我们促进很大。

(3) Thank you, iSolver users, for your online and offline communication, helping us find problems, providing feedback, and guiding us in the right direction. Perhaps we have little effect on you, but your opinions have greatly promoted us.

（4）　最后，也要感谢强大的CAE商软和理论大师们。在商软面前，我们真的微不足道，对有限元的理论发展毫无创新，仅仅只是笨拙的仿制和翻译。反而是商软给了我们一个考核的标准和工作上的目标，在深入商软内部实现方式的一个个征途中，当研究了很久的问题和商软对上的一刹那带给我们心灵的满足是其它东西无法代替的。

(4) Finally, we also want to thank the powerful CAE commercial software and the theoretical masters. In front of the commercial software, we are really insignificant, with no innovation in the development of finite element theory, we are just clumsy imitators and translators. Instead, the commercial software has given us a standard for assessment and a goal for work. In the journey of delving into the internal implementation of the commercial software, the moment when we have studied a problem for a long time and it aligns with the commercial software, it brings us a sense of satisfaction that nothing else can replace.

共同学习，共同进步。　Learn together, progress together.

如果有任何其它疑问，欢迎联系我们：　If you have any other questions, please feel free to contact us:

SnowWave02 From www.jishulink.com

email: snowwave02@qq.com