有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量

Theoretical Foundation of Finite Element Analysis and Research on the Internal Implementation of Abaqus Series 25: Stable Time Increment in Explicit Analysis

SnowWave02 关注 Focus

2020年7月12日 03:20 July 12, 2020 03:20 浏览：3995 Views: 3995 评论：15 Comments: 15 收藏：15 Favorites: 15

（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25： 显式分析的稳定时间增量的图1
有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图2

==概述== ==Overview==

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图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，通过

This series of articles studies the mature finite element theory foundation and its implementation in commercial finite element software, through

（1） 基础理论 (1) Basic Theory

（2） 商软操作 (2) Commercial Software Operation

（3） 自编程序 (3) Self-written program

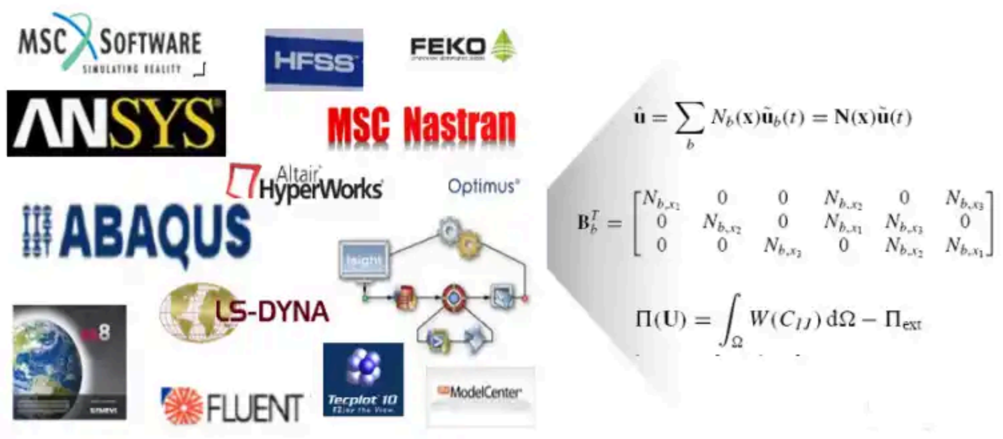
三者结合的方式将复杂繁琐的结构有限元理论通过简单直观的方式展现出来，同时深层次的学习有限元理论和商业软件的内部实现原理。

The combination of the three methods presents the complex and cumbersome structural finite element theory in a simple and intuitive way, while also deeply studying the internal implementation principles of finite element theory and commercial software.

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。

The theoretical development of finite elements 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 a black box to the outside, and users can only guess the internal implementation methods from outside, except for developers.



$$\hat{\mathbf{u}} = \sum_b N_b(\mathbf{x})\tilde{\mathbf{u}}_b(t) = \mathbf{N}(\mathbf{x})\tilde{\mathbf{u}}(t)$$

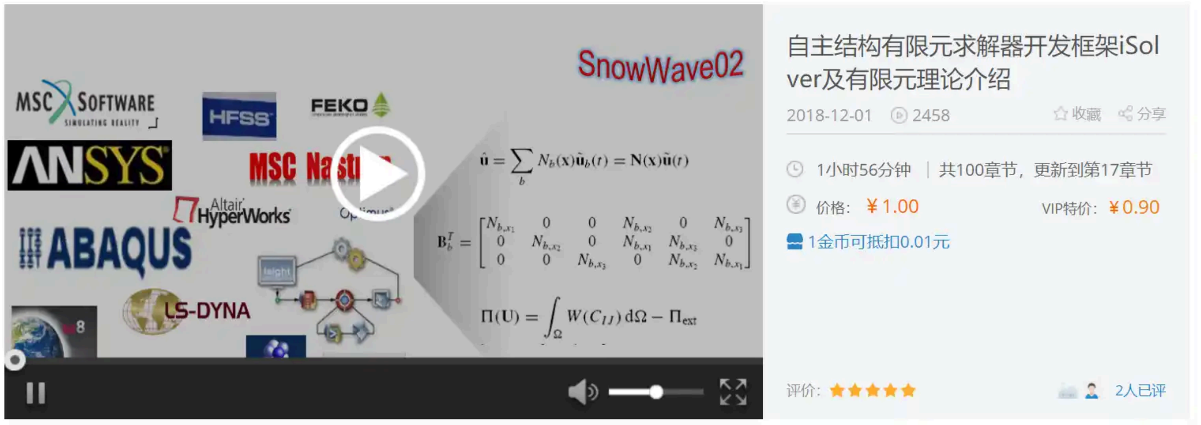
$$\mathbf{B}_b^T = \begin{bmatrix} N_{b,x_1} & 0 & 0 & N_{b,x_2} & 0 & N_{b,x_3} \\ 0 & N_{b,x_2} & 0 & N_{b,x_1} & N_{b,x_3} & 0 \\ 0 & 0 & N_{b,x_3} & 0 & N_{b,x_2} & N_{b,x_1} \end{bmatrix}$$

$$\Pi(\mathbf{U}) = \int_\Omega W(C_{IJ})\,d\Omega - \Pi_{ext}$$

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.

自主结构有限元求解器iSolver介绍视频： Introduction Video of Autonomous Structural Finite Element Solver iSolver

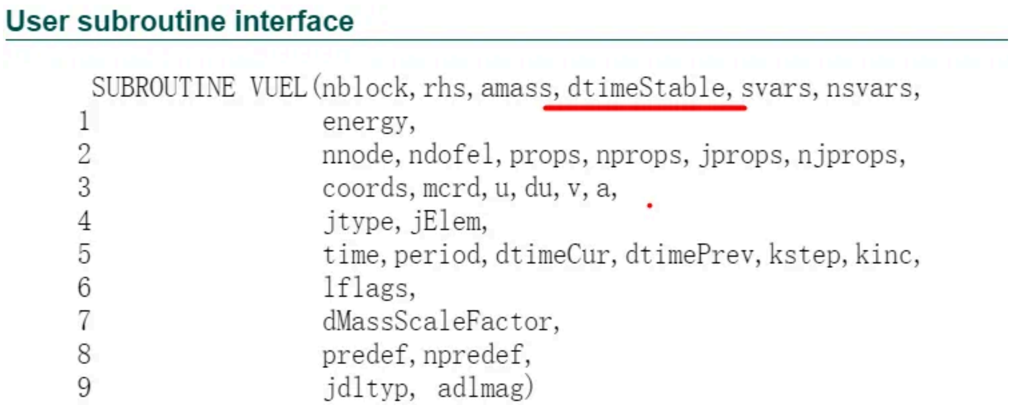


http://www.jishulink.com/college/video/c12884

==第25篇：显式分析的稳定时间增量== ==The 25th Article: Stable Time Increment in Explicit Analysis==

相对隐式分析，显式分析在单元函数中的计算要相对简单，只需计算应力的更新，而少了单元刚度矩阵的求解，但显式在增量步的自动步长、网格畸变的控制、质量缩放等方面又比隐式要多做许多工作。其中求解过程中的稳定时间增量是每个显式分析都会遇到的问题，由于没有迭代，显式分析都能得到某个结果，不存在收敛问题，而结果的正确性和稳定时间增量密切相关，很多人做显式分析一般直接交给商软去做自动步长，当商软显式结果发散或者结果差异很大时也不清楚怎么修改时间步长。同时，对于显式自定义单元VUEL等子程序，Abaqus要求用户自己计算稳定时间增量dtimestable，Abaqus会根据用户计算的稳定时间增量来限制它的增量步长。本文将简单介绍一下稳定时间增量的概念和理想及工程应用上的两种计算方式，并用Abaqus中一个简单的算例来验证工程上的稳定时间增量的计算公式，便于你对稳定时间增量的理解和自己编程实现。

Compared to implicit analysis, explicit analysis has simpler calculations in element functions, only requiring the update of stresses, without the need to solve the element stiffness matrix. However, explicit analysis has to do much more work in terms of automatic step length, control of mesh distortion, and mass scaling than implicit analysis. Among them, the stable time increment during the solution process is a problem that every explicit analysis encounters. Since there is no iteration, explicit analysis can always obtain some results, and there is no convergence problem. However, the correctness of the results is closely related to the stable time increment. Many people who do explicit analysis generally entrust the automatic step length to commercial software, and when the explicit results of the commercial software diverge or the results differ greatly, they are not clear on how to modify the time step. At the same time, for explicit custom elements, VUEL subroutines, etc., Abaqus requires users to calculate the stable time increment dtimestable themselves. Abaqus will limit its increment step length based on the stable time increment calculated by the user. This article will briefly introduce the concept of stable time increment and two calculation methods in ideal and engineering applications, and use a simple example in Abaqus to verify the calculation formula of the stable time increment in engineering, which is convenient for your understanding of the stable time increment and your own programming implementation.


User subroutine interface

```
      SUBROUTINE VUEL(nblock,rhs,amass,dtimeStable,svars,nsvars,
     1                energy,
     2                nnode,ndofel,props,nprops,jprops,njprops,
     3                coords,mcrd,u,du,v,a,
     4                jtype,jElem,
     5                time,period,dtimeCur,dtimePrev,kstep,kinc,
     6                lflags,
     7                dMassScaleFactor,
     8                predef,npredef,
     9                jdltyp, adlmag)
```

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图9

1.1.1 稳定性的含义 1.1.1 The Meaning of Stability

在本系列13篇：显式和隐式的区别中提到，显式分析都是条件稳定的，譬如下面求解一个微分方程：

As mentioned in the 13 articles in this series: Differences between Explicit and Implicit Analysis, explicit analysis is always conditionally stable. For example, the following differential equation is solved:

$$y'(x) = -y+x+1$$

$y'(x) = -y + x + 1$

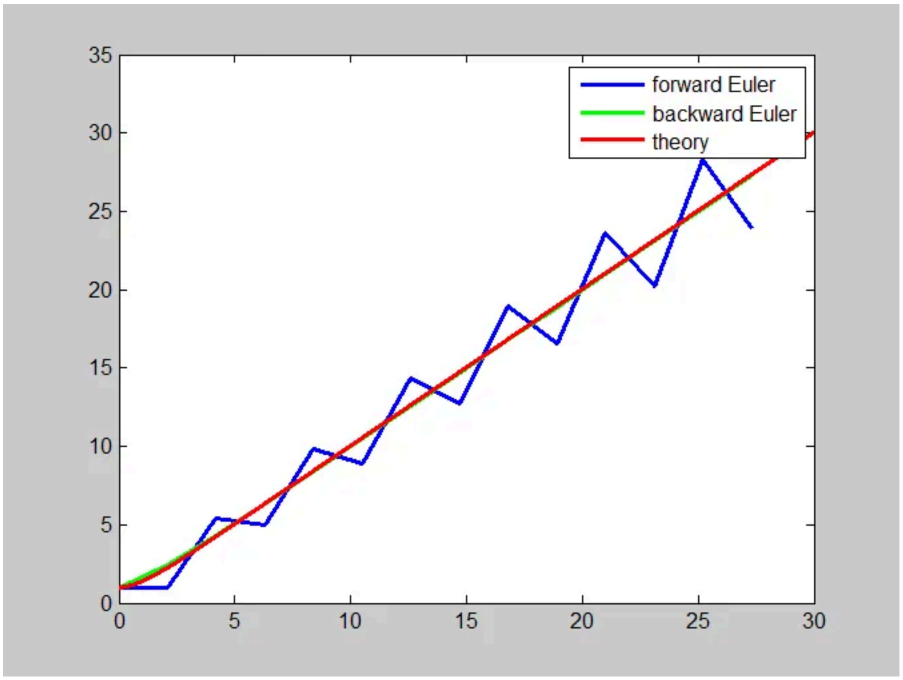
其中y(0) = 1。显式分析可得到下面的增量表达式：

Where y(0) = 1. The explicit analysis yields the following incremental expression:

$$y_{n+1} = (1-h)^n + O(s^2)$$

当h<2时，y收敛，但h>=2时，y将发散。当h=2.1时，可发现显式分析的蓝色线将和理论值越来越远。

When h < 2, y converges, but when h >= 2, y will diverge. When h = 2.1, it can be observed that the blue line of the explicit analysis will deviate more and more from the theoretical value.



上面针对的是一个具体的数学函数，对一个任意的有限元动力学系统，只要采用显式分析，也同样存在稳定性问题。动力学问题如果把质量导致的惯性力考虑在内，也是在解下面的平衡方程：

Above is a specific mathematical function, for an arbitrary finite element dynamic system, there is also a stability problem as long as explicit analysis is used. If the inertial force caused by mass is considered in the dynamic problem, it is also solving the following equilibrium equation:

$$K(t) * X(t) + M * X\ddot{(}t) - F(t) = 0$$

如果上式所有量都是t或者所有量都是t+dt，那么上式肯定是平衡的，没有一点问题。但在有限元求解过程中，只能通过已知时刻点t来求未知时刻点t+dt的值，而某些值在t+dt的值是不知道的，譬如刚度矩阵K，所以，只能退而求其次，上式中部分是时刻t的，譬如刚度矩阵K，而部分是时刻t+dt的值，譬如载荷F和质量阵M。

If all the quantities in the above equation are t or all are t+dt, then the equation is definitely balanced without any problem. However, in the finite element solution process, it is only possible to obtain the value of an unknown time point t+dt from the known time point t, and some values at t+dt are unknown, such as the stiffness matrix K. Therefore, one has to settle for second best, where part of the equation is at time t, such as the stiffness matrix K, while part is at time t+dt, such as the load F and the mass matrix M.

$$K(t) * X(t+dt) + M * \ddot{X}(t+dt) - F(t+dt) = Tol \ \neq 0$$

这么取值后上式左边就不再=0了，产生了人为导致的误差。而这种人为的误差，如果是隐式分析，那么可以通过迭代来解决，迭代多次后最后的时刻的Tol可以达到预定的小量，方程依然是平衡的。但对显式分析，流程如下所

示，不会再次迭代，只会进行下一次的t+2*dt的求解，这样，就可能导致两种结果：

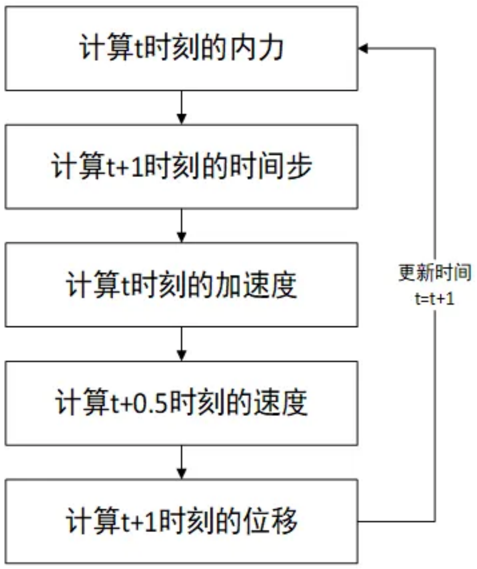
With such a choice, the left side of the equation is no longer equal to 0, resulting in an artificial error caused by human factors. For implicit analysis, this kind of artificial error can be solved through iteration, and after multiple iterations, the final time's Tol can reach the predetermined small amount, and the equation remains balanced. However, for explicit analysis, the process is as follows, and there will be no further iteration; it will only proceed to the next calculation of t+2*dt, which may lead to two different results:

（1） t时刻的误差将在t+dt时刻累积，此时系统就是不稳定的。譬如上图中的蓝色曲线。

The error at time t will accumulate at time t+dt, indicating an unstable system. For instance, as shown in the blue curve in the figure above.

（2） t时刻的误差在t+dt时刻不累积，此时系统就是稳定的。譬如上图中的如果h<2。

The error at time t does not accumulate at time t+dt, indicating a stable system. For instance, as shown in the figure above if h<2.



有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图18

1.1.2 稳定时间增量的理想计算方式 1.1.2 Ideal Calculation Method for Stable Time Increment

总结显式分析中增量步之间的关系，可以发现增量步和上一个增量步有简单的关系：

Summarizing the relationship between increment steps in explicit analysis, it can be found that there is a simple relationship between the increment step and the previous increment step:

$$X(n*dt) = A*X\big((n-1)*dt\big) + B$$

那么当前时刻和第一个时刻的一个简单关系： Then, the simple relationship between the current moment and the first moment:

$$X(n*dt) = C^n * X(0) + D$$

而这个C类似于一个比例系数，与dt的取值有关，譬如上述函数例子中C=1-dt。对这种只有一个自由度的问题，可以发现|C|<1时，系统稳定，否则系统不稳定。在数学上可以证明C与系统的特征值是密切相关的。也就是说dt与系统的特征值有关，对一个有限元系统来说，数学上已经证明dt如果满足下式就是稳定的：

This C is similar to a proportionality coefficient, which is related to the value of dt, for example, in the above function example, C=1-dt. For problems with only one degree of freedom, it can be found that when |C|<1, the system is stable, otherwise the system is unstable. Mathematically, it can be proven that C is closely related to the eigenvalues of the system. That is, dt is related to the eigenvalues of the system, and mathematically, it has been proven that dt is stable if it satisfies the following formula:

$$dt < \frac{2}{\omega_m} = dt\_ideal$$

其中 Among

$$\omega_m$$

为有限元系统的最大特征值。我们称右端为稳定时间增量的理想计算方式。

The maximum eigenvalue of the finite element system. We call the right-hand side the ideal calculation method of the stable time increment.

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图25

1.1.3 稳定时间增量简化的必要条件 1.1.3 Necessary Conditions for Simplification of the Stable Time Increment

在稳定时间增量dt_ideal的理想计算方式中，需要计算有限元系统的最大特征值。而如果每次显式分析前都要计算一遍最大特征值，且对几何变化大的问题，在每个增量步都要更新一次，那么计算量将非常大。更何况对非对称的刚度等系统，还没有办法得到特征值。在实际有限元的工程计算中，会找另一种简单的稳定时间增量dt_engeer的工程计算方式，这个dt_engeer必须满足两个条件：

In the ideal calculation method of the stable time increment dt_ideal, it is necessary to calculate the maximum eigenvalue of the finite element system. If the maximum eigenvalue needs to be recalculated before each explicit analysis, and for problems with large geometric changes, it needs to be updated at each increment step, the computational effort will be very large. Moreover, for systems with asymmetric stiffness and the like, it is not possible to obtain the eigenvalues. In actual finite element engineering calculations, another simple engineering calculation method for the stable time increment dt_engineer is used, which must satisfy two conditions:

（1） 条件1：这种计算方式并不一定严格求出上面的dt_ideal，但求出的稳定时间一定比理想的稳定时间增量更小。即当有限元的时间增量步长dt<dt_engeer，必然有dt<dt_ideal，此时系统依然满足稳定性要求。

(1) Condition 1: This calculation method does not necessarily strictly calculate the above dt_ideal, but the calculated stable time is always smaller than the ideal stable time increment. That is, when the finite element time increment step dt < dt_engineer, it is always true that dt < dt_ideal, and the system still meets the stability requirements.

（2） 条件2：dt_engeer比dt_ideal不能小太多，应该是同一量级，否则增量步将大幅增加，导致系统耗时过大。

(2) Condition 2: dt_engine should not be much smaller than dt_ideal; they should be of the same order of magnitude. Otherwise, the increment step will increase significantly, leading to excessive system time consumption.

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图26

1.1.4 一般有限元的稳定时间增量的工程计算方法

1.1.4 Engineering calculation methods for the stability time increment of general finite elements

本着这个原则，一般有限元上做了两次计算简化工作：

Adhering to this principle, generally, two simplification steps are performed in finite element analysis:

（1）第一步：是将整个系统的最大特征值取为所有单个单元的特征值,由于系统的约束会压缩总体频率，导致dt_element<dt_ideal。也就是：

(1) The first step is to take the maximum eigenvalue of the entire system as the eigenvalue of all individual elements, as the constraints of the system will compress the overall frequency, resulting in dt_element < dt_ideal. That is to say:

$$dt < \frac{2}{\omega_m^e} = \mathrm{dt\_element}$$

（2）单个单元依然有特征值求解问题，因此，再进一步简化，实际取的稳定时间增量不再需要计算特征值，而是估算为：

(2) There is still an eigenvalue solving problem for individual elements, therefore, further simplification is made, and the actual stable time increment taken does not need to calculate the eigenvalue but is estimated as:

$$dt < min(L_e/C_d) = dt\_engeer$$

其中min表示对所有单元遍历后的最小值，Le为单元特征长度，Cd为材料疏密波速度，对杆件有Cd=sqrt(E/d), E、d分别为杆长度、杨氏模量和密度。按此公式计算将非常简单，只需遍历所有单元，将得到的参数代入上式即可。

In which min represents the minimum value after traversing all elements, Le is the element eigenlength, Cd is the material density wave velocity, for rods, Cd = sqrt(E/d), where E and d are the length, Young's modulus, and density of the rod, respectively. Calculating according to this formula will be very simple, just need to traverse all elements and substitute the obtained parameters into the formula.

为了验证上述简化满足前面所述两点，我们举一个由杆单元truss组成的简单系统来验证一下dt_engeer的具体值。此时，每个杆单元的频率可以简单的估算为

To verify that the above simplification meets the two points mentioned earlier, we take a simple system composed of bar element trusses to verify the specific value of dt_engeer. At this time, the frequency of each bar element can be simply estimated as

$$\omega_i = \sqrt{\frac{K}{m}} = \sqrt{\frac{EA/L}{0.5 * d * A * L}} = \frac{1}{L}\sqrt{\frac{E}{0.5d}}$$

其中L、E、d分别为杆长度、杨氏模量和密度。注意上式我们用了集中质量。此时的第一步稳定时间增量

Among them, L, E, and d represent the rod length, Young's modulus, and density, respectively. Note that we used a concentrated mass in the above formula. The first step of the stable time increment at this point

$$dt\_element = \frac{2}{\omega_m^e} = \frac{1.414 * L}{\sqrt{\frac{E}{d}}}$$

显然，dt_engeer和dt_element同一量级，且略小。

It is obvious that dt_engeer and dt_element are of the same order of magnitude, and slightly smaller.

由于dt_engeer<dt_ideal，可以发现有限元商软采用的dt_engeer不一定是最优的，如果你能找到一个时间增量步dt，使得dt_engeer<dt<dt_ideal，那么可以比商软默认的耗时更少。后面的例子中也证明了这点。

Since dt_engeer < dt_ideal, it can be found that the dt_engeer adopted by the finite element commercial software may not be the optimal one. If you can find a time increment step dt such that dt_engeer < dt < dt_ideal, then it can be more time-efficient than the default time increment provided by the software. This point is also demonstrated in the following examples.

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图35

1.1.5 Abaqus对稳定时间增量的工程计算方法的修正

1.1.5 Correction of the Engineering Calculation Method for Stable Time Increment in Abaqus

Abaqus对稳定时间增量的工程计算还做了两处不同的修正：

Abaqus has also made two different corrections to the engineering calculation of the stable time increment.

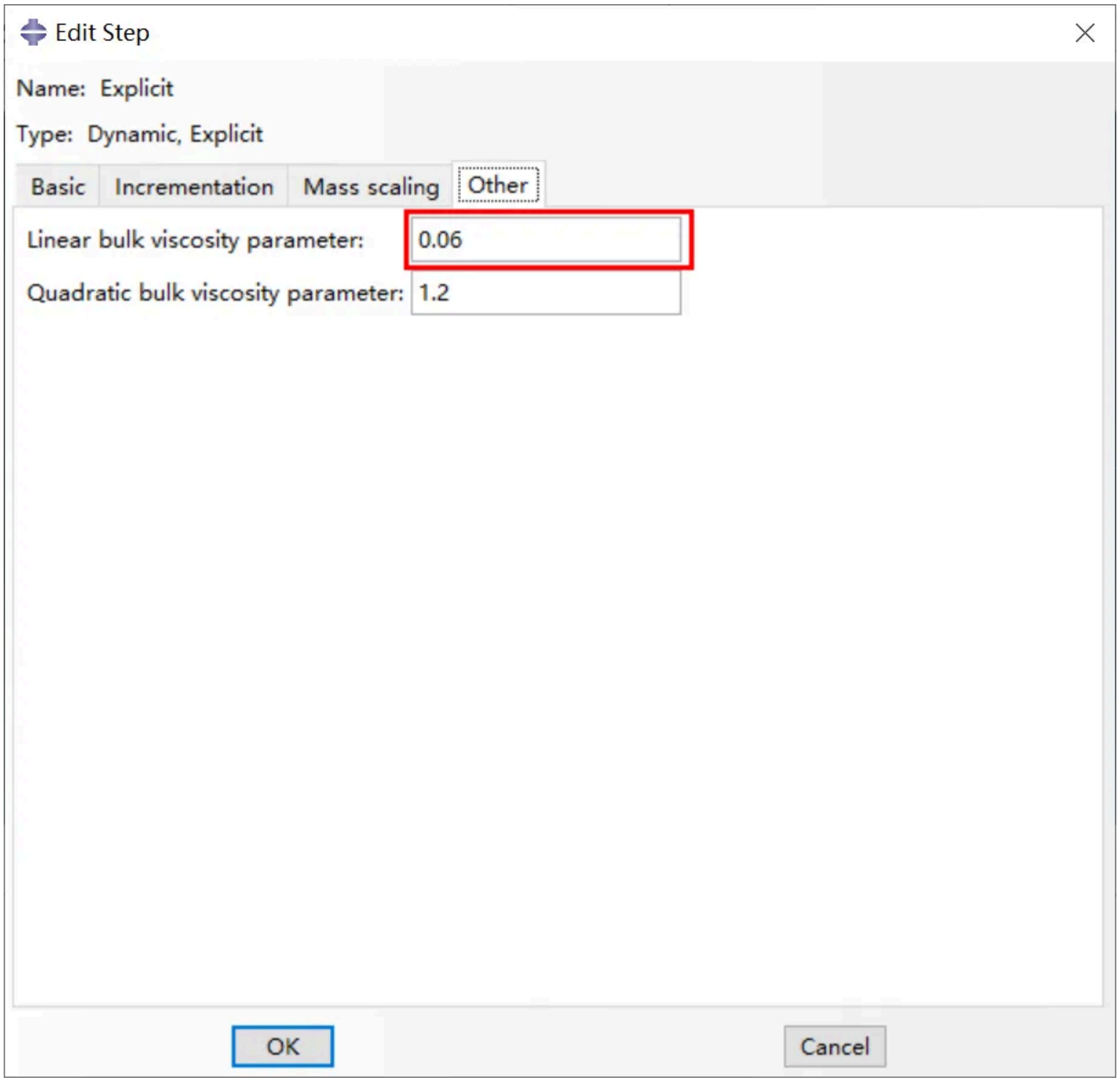
（1） 考虑Bulk viscosity效应修正工程稳定时间增量,主要是利用Linear Bulk Viscosity系数b1减小dt_engeer。即

(1) Consider the correction of the engineering stable time increment due to Bulk viscosity effect, mainly by using the Linear Bulk Viscosity coefficient b1 to reduce dt_engeer.

$$dt < min(L_e/C_d) * (\sqrt{1 + b1^2} - b1)$$

b1在Step->Other的Linear Bulk Viscosity可修改。

b1 can be modified in the Linear Bulk Viscosity under Step->Other.



Name: Explicit

Type: Dynamic, Explicit

Linear bulk viscosity parameter: 0.06

Quadratic bulk viscosity parameter: 1.2

默认为0.06，此时 Default is 0.06, at this time

$$(\sqrt{1+b1^2}-b1=0.9418$$

（2） 为了避免系统对实数的精度或者截断误差，加了一个Tolerance，使得

(2) In order to avoid the system's precision issues or truncation errors for real numbers, a Tolerance has been added

$$dt < min\left(\frac{L_e}{C_d}\right) * \left(\sqrt{1+b1^2}-b1\right) * (1-Tolerance) ----Final$$

显然这个Tolerance是个远小于1的值，Abaqus取为0.01，

It is obvious that this Tolerance is a value much less than 1, and Abaqus takes it as 0.01,

- 此值我们没发现在Abaqus界面上怎么修改，如果谁能找到，也希望能交流一下。

  This value does not seem to be able to be modified on the Abaqus interface, and if anyone finds it, I hope we can communicate as well.

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图44 1.2 Abaqus的实现验证 1.2 Verification of Abaqus Implementation

我们将在Abaqus中采用一个简单的显式分析算例，来验证两个问题：

We will use a simple explicit analysis example in Abaqus to verify two issues:

（1） Abaqus采取的稳定时间增量和上述最后的稳定时间增量的计算公式一致。

(1) The stable time increment adopted by Abaqus is consistent with the calculation formula of the last stable time increment mentioned above.

（2） 对于某些问题，其实Abaqus采取的稳定时间增量是dt_engeer，相对保守，实际上可以取一个更加接近dt_ideal的值，系统依然是稳定的。

(2) For certain problems, in fact, the stable time increment adopted by Abaqus is dt_engeer, which is relatively conservative. In reality, a value closer to dt_ideal can be chosen, and the system remains stable.

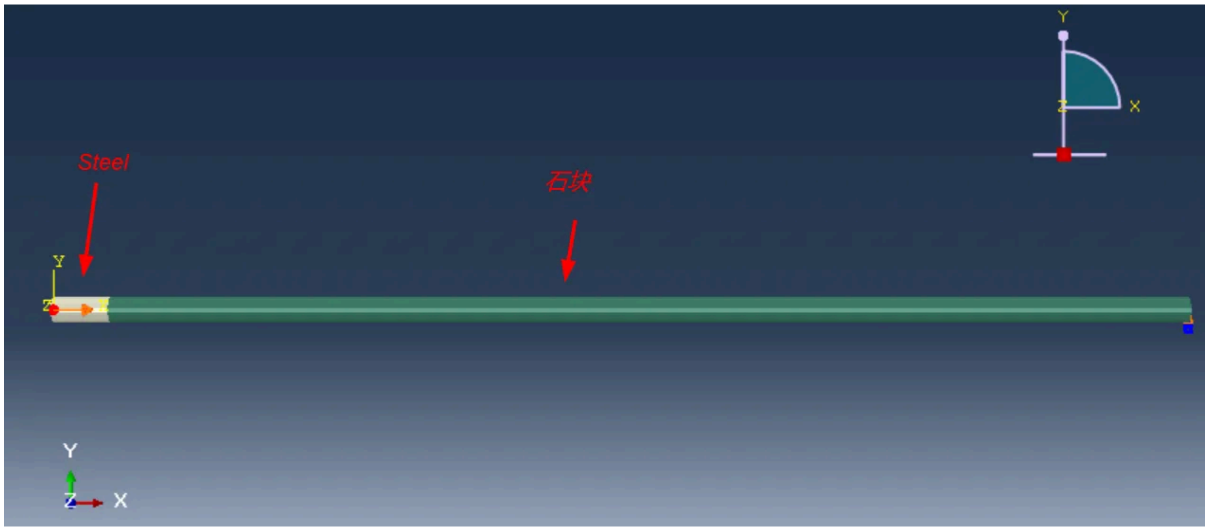
有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图45 1.2.1 模型例子 1.2.1 Model Example

模型为一根长1m，截面为4平方cm的圆柱，左端5cm由Steel组成，杨氏模量和密度分别是2e11Pa和7800千克每立方米，剩下部分由碎石组成，杨氏模量和密度分别是4.432e6Pa和1560千克每立方米。右端点固定在墙上，左端点面载荷4e6sin(150t)。我们只研究0-0.01s时间内的位移。

The model is a cylinder with a length of 1m and a cross-sectional area of 4 square cm. The left 5cm is made of steel with a Young's modulus and density of 2e11 Pa and 7800 kg/m^3, respectively. The remaining part is made of gravel with a Young's modulus and density of 4.432e6 Pa and 1560 kg/m^3. The right end is fixed to a wall, and the left end is subjected to a surface load of 4e6sin(150t). We only study the displacement within the time interval of 0-0.01s.

在Abaqus中建模如下，我们简单将模型划分为20个单元。采用truss单元。

In Abaqus, the model is built as follows, where we simply divide the model into 20 elements. Truss elements are used.



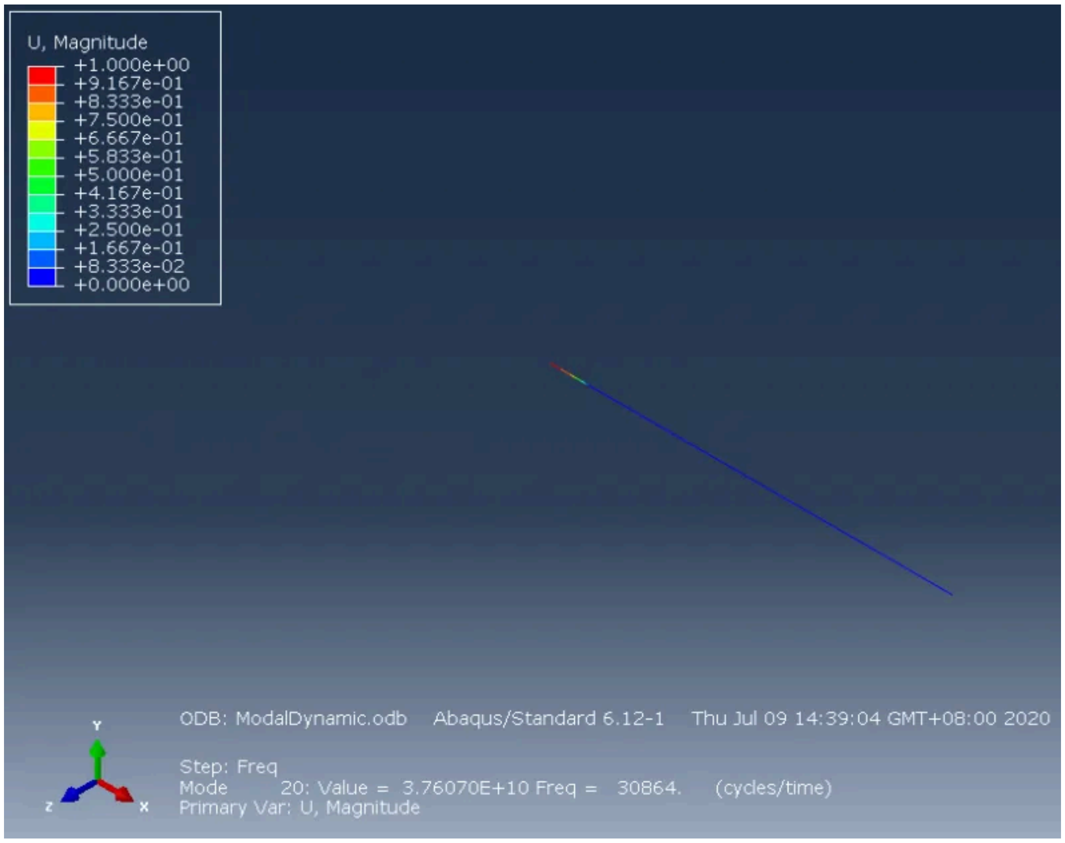
有限元理论基础及Abaqus内部实现方式研究系列25：显式分析的稳定时间增量的图48

1.2.2 稳定时间增量的理论值 1.2.2 Theoretical value of stable time increment

1.2.2.1 稳定时间增量的理想计算的理论值 1.2.2.1 Ideal calculation of the theoretical value of the stable time increment

理想计算方式需要先计算系统最大模态特征，由于是20个单元，采用truss单元，就相当于只有21个自由度，右端约束后，无约束的自由度为20个，得到的K和M矩阵的秩为20，那么无论用哪种模态计算方法，得到的模态最大为20阶。在Abaqus中计算，结果如下，可得20阶的模态频率为30864Hz。

The ideal calculation method requires the calculation of the maximum modal eigenvalue of the system first. Since there are 20 elements, using truss elements is equivalent to having only 21 degrees of freedom. After the right end is constrained, there are 20 degrees of freedom without constraints, and the rank of the K and M matrices obtained is 20. Therefore, regardless of which modal calculation method is used, the maximum modal order obtained is 20. In Abaqus, the calculation results are as follows, and the modal frequency of the 20th order can be obtained as 30864Hz.



ODB: ModalDynamic.odb   Abaqus/Standard 6.12-1   Thu Jul 09 14:39:04 GMT+08:00 2020

Step: Freq
Mode 20: Value = 3.76070E+10 Freq = 30864. (cycles/time)
Primary Var: U, Magnitude

$$dt\_ideal = \frac{2}{\omega_{20}} = \frac{2}{30864 * 2 * \mathrm{pi}} = 1.0315e - 5$$

1.2.2.2 稳定时间增量的工程计算的理论值 1.2.2.2 Theoretical value of engineering calculation for stable time increment

最小的工程稳定时间增量显然是左端的Steel单元，此时为：

The minimum engineering stable time increment is obviously the Steel element on the left, at this time:

$$dt\_engeer = \left(\frac{L_e}{C_d}\right) * \left(\sqrt{1 + b1^2} - b1\right) * (1 - Tolerance) = \frac{0.05}{\sqrt{\frac{2e11}{7800}}} * 0.9418 * 0.99$$

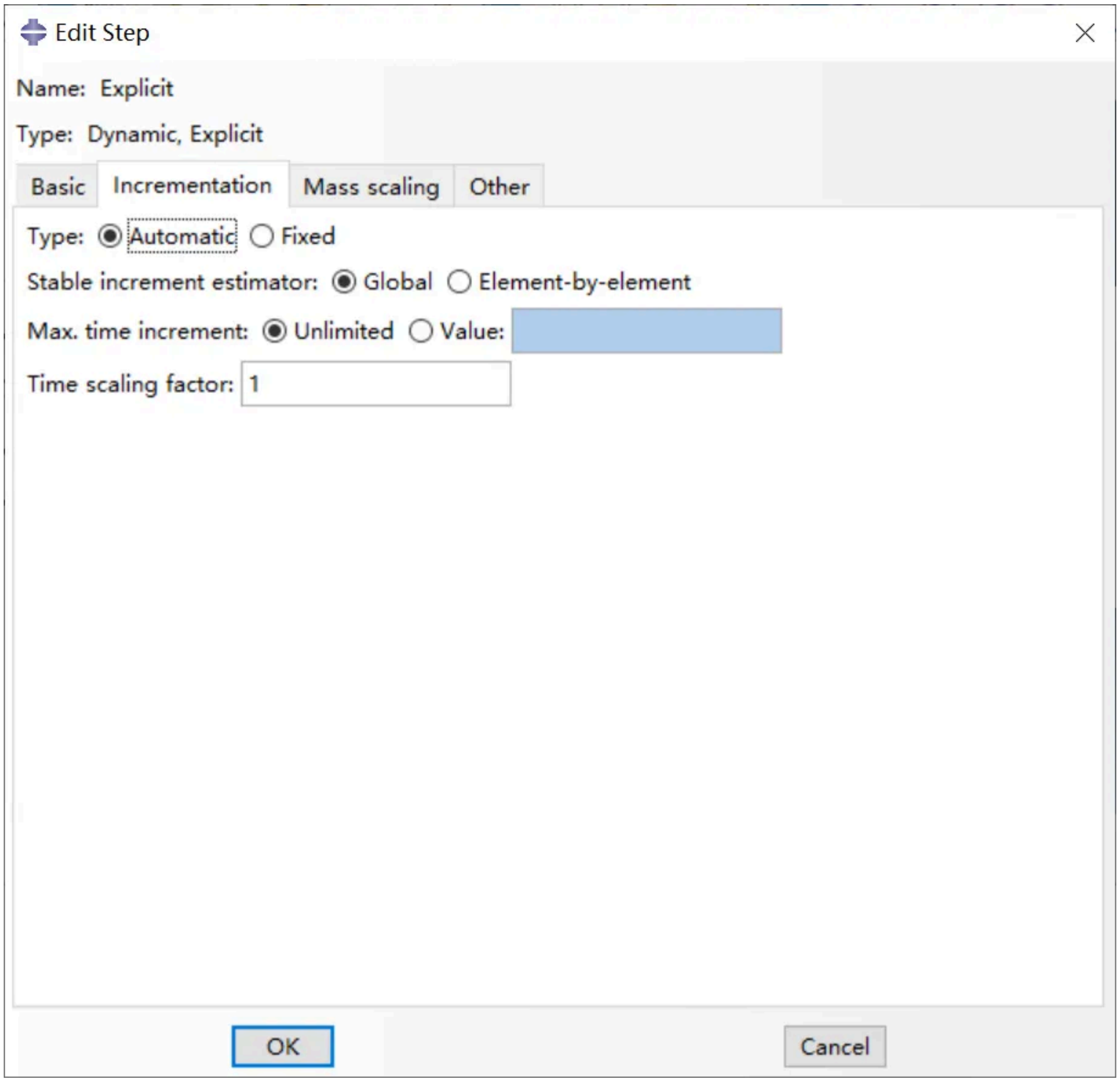
$$= 9.206518768647146e - 006$$

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图54

1.2.3 自动步长 1.2.3 Automatic Step Size

在Abaqus中选择显式分析，dynamic,explicit，同时设置为自动步长。

In Abaqus, select explicit analysis, dynamic, explicit, and set it to automatic time step.


Edit Step

Name: Explicit

Type: Dynamic, Explicit

Type: [✓ Automatic] Fixed

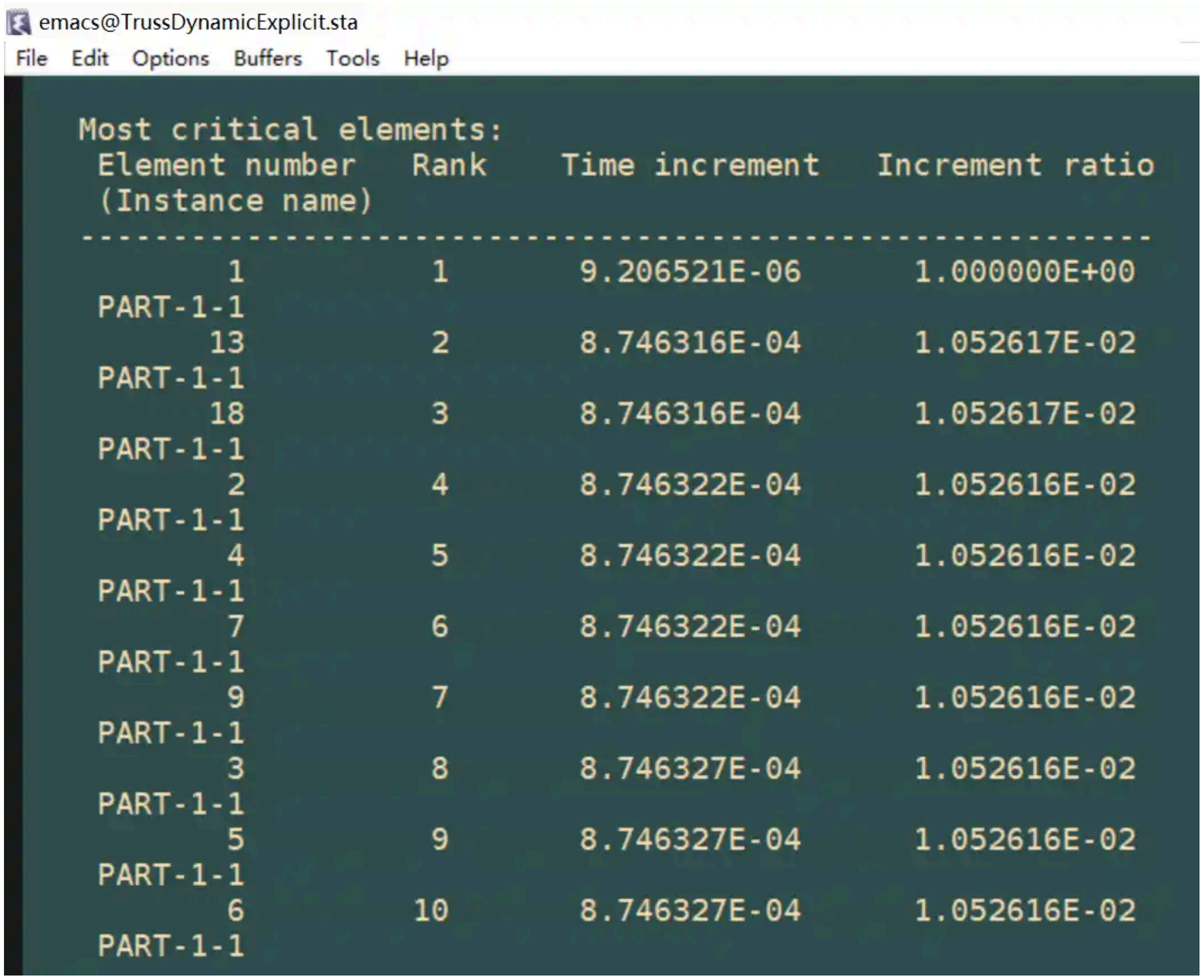
Stable increment estimator: [✓ Global] Element-by-element

Max. time increment: [✓ Unlimited] Value:

Time scaling factor: 1

运行结束后查看.sta文件，Abaqus会在此文件中在第一个增量前记录前十个最小的单元稳定时间增量。可发现如下所示，最小的单元稳定时间增量为第一个单元，且值和理论完全一致：

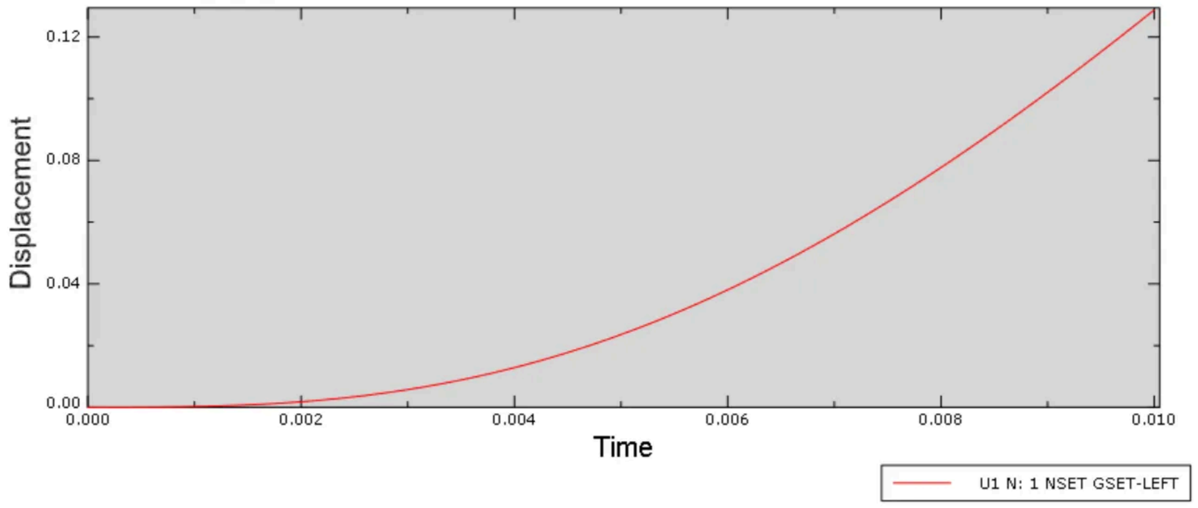
After the run, check the .sta file. Abaqus will record the first ten smallest element stable time increments before the first increment in this file. It can be found as shown below: the smallest element stable time increment is the first element, and the value is completely consistent with the theory.


emacs@TrussDynamicExplicit.sta
File Edit Options Buffers Tools Help

```
Most critical elements:
 Element number   Rank    Time increment   Increment ratio
 (Instance name)
---------------------------------------------------------
       1           1       9.206521E-06      1.000000E+00
 PART-1-1
      13           2       8.746316E-04      1.052617E-02
 PART-1-1
      18           3       8.746316E-04      1.052617E-02
 PART-1-1
       2           4       8.746322E-04      1.052616E-02
 PART-1-1
       4           5       8.746322E-04      1.052616E-02
 PART-1-1
       7           6       8.746322E-04      1.052616E-02
 PART-1-1
       9           7       8.746322E-04      1.052616E-02
 PART-1-1
       3           8       8.746327E-04      1.052616E-02
 PART-1-1
       5           9       8.746327E-04      1.052616E-02
 PART-1-1
       6          10       8.746327E-04      1.052616E-02
 PART-1-1
```

此时总共增量步为1087次，得到的左端位移随时间的变换曲线如图：

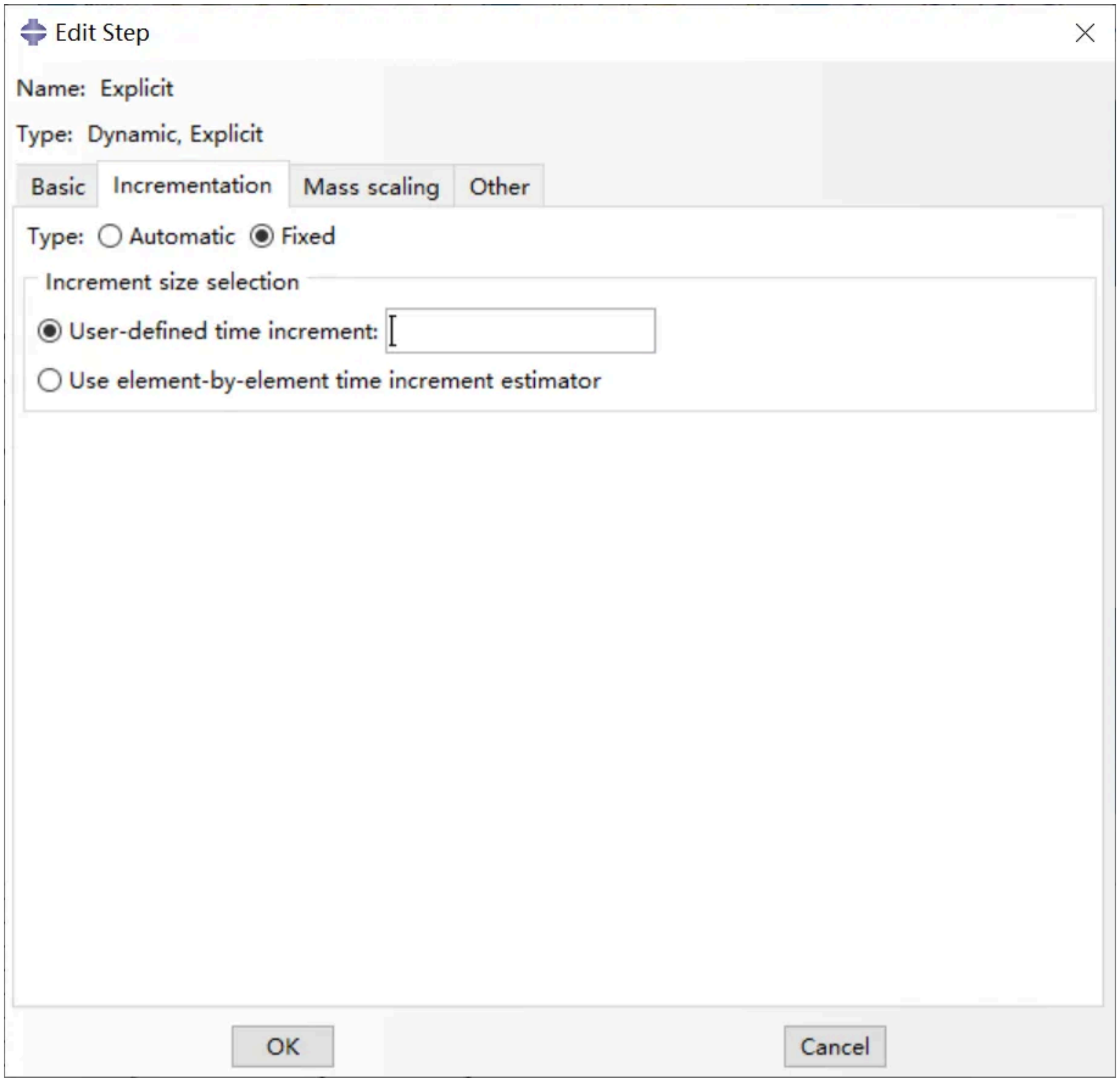
At this time, the total number of increment steps is 1087, and the displacement at the left end as a function of time is shown in the figure:



有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图60

1.2.4 固定步长 1.2.4 Fixed Step Size

Abaqus中改为固定步长： Change to fixed step in Abaqus:



Name: Explicit

Type: Dynamic, Explicit

Type: Automatic [✓ Fixed]

Increment size selection

[✓ User-defined time increment:]

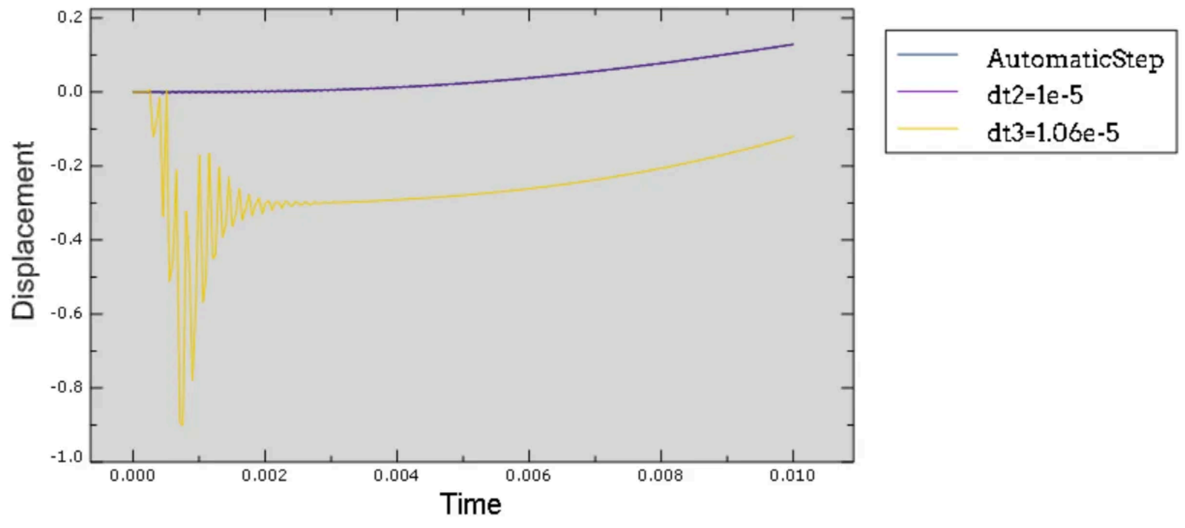
Use element-by-element time increment estimator

取固定步长分别为dt2=1e-5和dt3=1.06e-5，即

Take fixed time steps of dt2=1e-5 and dt3=1.06e-5, respectively.

dt_engeer<dt2<dt_ideal<dt3

得到的左端位移随时间变化曲线如下： The curve of the displacement at the left end varies with time as follows:



可发现dt3已经发散，而dt2和自动步长基本一致，但dt2只计算了1000个增量步，比自动步长少了8.7%。从而验证了，对某些问题，为了加速，时间增量可以取一个超过Abaqus默认更加接近dt_ideal的值，系统依然是稳定的。

It can be observed that dt3 has diverged, while dt2 and the automatic step size are basically consistent. However, dt2 only calculates 1000 increments, which is 8.7% less than the automatic step size. This verifies that for certain problems, in order to accelerate, the time increment can be taken to be a value exceeding the Abaqus default and closer to dt_ideal, and the system remains stable.

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图65
1.3 联系方式 1.3 Contact Information

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25： 显式分析的稳定时间增量的图66
有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图67
有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图68

以往的系列文章： Previous series articles:

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图69
有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图70

1.4.1 ========第一阶段========

1.4.1 ========First Phase========

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。

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。

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：**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

有限元理论基础及Abaqus内部实现方式研究系列25：显式分析的稳定时间增量的图71
有限元理论基础及Abaqus内部实现方式研究系列25：显式分析的稳定时间增量的图72

1.4.2 ========第二阶段========

1.4.2 ========Second Stage========

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

第二十篇：**UEL用户子程序开发步骤。**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。

第二十篇：UEL 用户子程序开发步骤。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 flexible.

https://www.jishulink.com/content/post/1204261

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图73

有限元理论基础及Abaqus内部实现方式研究系列25： 显式分析的稳定时间增量的图74

1.4.3 ========第三阶段========

1.4.3 ========Third Phase========

第二十一篇：**自主CAE开发实战经验第二阶段总结。**从实战角度介绍自主CAE在推广和工程化应用的过程中的体会，同时说明一个CAE平台最重要的两个特点：可扩展和易维护。

Chapter 21: Summary of the Second Stage of Autonomous CAE Development Practice. Introduce the experience in promoting and engineering the application of autonomous CAE from a practical perspective, and at the same time, explain the two most important characteristics of a CAE platform: scalability and ease of maintenance.

https://www.jishulink.com/content/post/1204970

第二十二篇：**几何非线性的刚度矩阵求解。**介绍几何非线性下的刚度矩阵的理论推导和计算机求解方法，最后利用一个简单的算例验证我们对Abaqus几何非线性的刚度矩阵的实现方式的猜测。

Chapter 22: Solution of Stiffness Matrix under Geometric Nonlinearity. Introduce the theoretical derivation and computer solution method of the stiffness matrix under geometric nonlinearity, and finally verify our guess about the implementation method of Abaqus geometric nonlinearity stiffness matrix through a simple example.

http://www.jishulink.com/content/post/1254435

第二十三篇：

有限元理论基础及Abaqus内部实现方式研究系列25：显式分析的稳定时间增量的图75

编写简单面内拉伸问题UEL Step By Step。通过简单面内拉伸问题UEL的编写，介绍解决有限元问题从理论到算法再到编程实现的一般流程以及面内拉伸问题的基本算法步骤，最后将UEL与Abaqus的S4R单元计算结果进行对比，进行验证。

Chapter 23: Step by Step Writing a Simple In-Plane Tension Problem UEL. By writing a simple in-plane tension problem UEL, this article introduces the general process of solving finite element problems from theory to algorithm to programming implementation, as well as the basic algorithmic steps of in-plane tension problems. Finally, the UEL is compared with the S4R element calculation results of Abaqus for verification.

http://www.jishulink.com/content/post/1256835

第二十四篇：

有限元理论基础及Abaqus内部实现方式研究系列25：显式分析的稳定时间增量的图76

显式求解Step By Step。本文概要性地介绍了Abaqus中心差分法的理论以及算法实现的整体流程，并通过简单弹簧显式动力学分析算例与iSolver和Abaqus计算结果进行对比，验证了算法和整体流程的正确性。

Chapter 24: Explicit Solution Step by Step. This article briefly introduces the theory of Abaqus central difference method and the overall process of algorithm implementation, and compares the results of a simple spring explicit dynamic analysis example with those of iSolver and Abaqus to verify the correctness of the algorithm and the overall process.

https://www.jishulink.com/content/post/1261165

推荐阅读 Recommended Reading