有限元理论基础及Abaqus内部实现方式研究系列4：非线性问题的求解

Theoretical Foundation of Finite Element Method and Research on Internal Implementation of Abaqus Series 4: Solution of Nonlinear Problems

SnowWave02

更新于2021年3月28日 12:49 Updated on March 28, 2021, 12:49

浏览: 5102 Views: 5102

评论: 28 Comments: 28

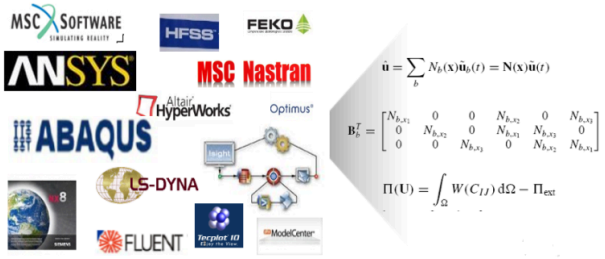
收藏: 33 Favorited: 33

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

1 概述 1 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 elements, but rather the implementation methods of commercial finite element software. The theoretical development of finite elements has matured for decades and commercial finite element software also adopts these mature finite element theories. However, in the process of practical application, 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 main commercial software manual will specify which theoretical formula each element uses, but only mention the method of correction, 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甚至是做自己的软件的朋友有些帮助。由于水平有限，里面可能有许多错

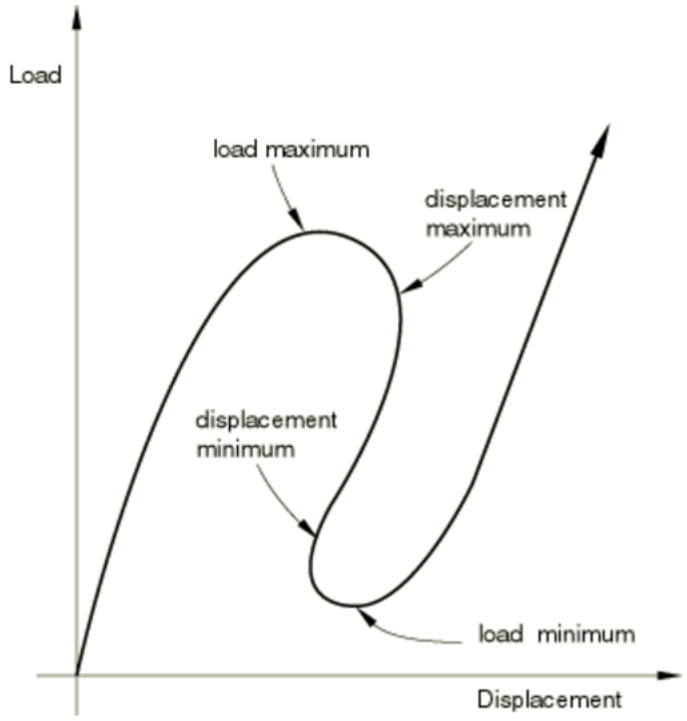
误，欢迎交流讨论。

On the one hand, we consult the Abaqus software manual for the description of the correction method, on the other hand, we program a simple structural finite element solver ourselves, and compare the results of our self-developed solver with Abaqus, combining theoretical manuals to study Abaqus' correction methods as if we were peering through a bamboo tube. This allows us to guess the internal calculation methods of commercial finite element software. While conducting research, 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 or even develop their own software. Due to my limited abilities, there may be many errors, and I welcome discussions and exchanges.

2 第四篇：非线性问题的求解 2 Fourth Part: Solution of Nonlinear Problems

随着分析对象越来越复杂，有限元软件从线性分析向非线性分析（如材料为非线性、几何大变形导致的非线性、接触行为引起的边界条件非线性等）发展，Abaqus的强大主要就在于能解决各种工程上的非线性问题。由于非线性理论和编程实现的困难性，研究起来也相对困难，只能一步步来，我们在本文中首先研究各种非线性都会遇到的求解问题和Abaqus的内部实现方式。很多人做非线性编程结果正确性验证由于不知道商软的内部实现方式，采用小模型或者有理论解的模型对比理论结果和商软最终的迭代结果，可能最终的结果差不多，但这种方法很容易就忽略了非线性中细节问题，如果进一步对比迭代过程，就会发现迭代效率和精度和商软相比差距较大，这种效率和精度当遇到复杂工程问题时就尤为重要了。我们试图通过自编程序和Abaqus非线性的每个迭代步的中间输出量进行对比，最终结合帮助文档猜测Abaqus软件非线性迭代的内部实现方法。

With the increasing complexity of the analysis object, finite element software has developed from linear analysis to nonlinear analysis (such as nonlinear materials, geometric large deformation-induced nonlinearities, and boundary condition nonlinearities caused by contact behaviors), and the strength of Abaqus mainly lies in its ability to solve various nonlinear engineering problems. Due to the difficulty of nonlinear theory and programming implementation, research on it is relatively difficult as well, and it can only be done step by step. In this article, we first study the solution problems encountered by various nonlinearities and the internal implementation methods of Abaqus. Many people do the correctness verification of nonlinear programming results due to not knowing the internal implementation methods of commercial software, and use small models or models with theoretical solutions to compare the theoretical results with the final iterative results of commercial software. Although the final results may be similar, this method is prone to ignore the detailed problems in nonlinearities. If further comparison is made with the iterative process, it will be found that the iteration efficiency and accuracy are much lower than that of commercial software. This efficiency and accuracy is particularly important when encountering complex engineering problems. We attempt to compare the intermediate output quantities of the nonlinear iterative steps of Abaqus with our self-written programs, and finally guess the internal implementation method of Abaqus software's nonlinear iteration by combining the help documentation.



2.1 非线性平衡方程及求解理论 2.1 Theory of Nonlinear Equilibrium Equations and Solution Methods

在非线性分析过程中，无论哪一类非线性问题，经过有限元离散后，都可以转化为求解一个非线性方程组，对一个简单的一维问题，最终得到的非线性平衡方程如下

In the process of nonlinear analysis, regardless of which type of nonlinear problem, after finite element discretization, it can be transformed into solving a set of nonlinear equations. For a simple one-dimensional problem, the final nonlinear equilibrium equation is as follows

$$K_T(u)u = \mathrm{P}$$

已知P，求u。如果是线性问题，那么直接用 Given P, find u. If it is a linear problem, then it can be directly used

$$\mathrm{u} = K_T^{-1}P$$

如果是非线性问题，K与u相关，那么就不能简单的用上面公式了。非线性问题求解有多种方法主要分为以下几类：增量法、迭代法、增量迭代法和弧长法，下面将具体讲一下这几类方法。

If it is a nonlinear problem, K is related to u, then the above formula cannot be used simply. There are various methods for solving nonlinear problems, mainly divided into the following categories: incremental method, iterative method, incremental-iterative method, and arc-length method. The following will specifically discuss these methods.

2.1.1 增量法 2.1.1 Incremental Method

增量法的基本思想就是将施加外载荷的过程分割为若干个增量步，每一步只施加一个比较小的载荷增量，总载荷引起的位移就是所有增量步位移增量的累加。

The basic idea of the incremental method is to divide the process of applying external loads into several incremental steps, with each step applying only a relatively small load increment. The total displacement caused by the total load is the sum of the displacement increments in all the incremental steps.

对于每一个载荷增量

，平衡方程 For each load increment, the equilibrium equation

$$K_T(u)\Delta u = \Delta P$$

在一个增量步内，我们可以把位移和载荷的变化看作是线性变化的过程。只要这些增量步取得够小，最后求解的位移结果与实际值之间的差异将在容许的误差范围内。

Within an incremental step, we can consider the changes in displacement and load as a linear process. As long as these incremental steps are sufficiently small, the difference between the final displacement solution and the actual value will be within the allowable error range.

理论上讲，如果没有极值点，也就是一个P和u一一对应，那么增量法总能求得真实解。但实际工程应用情况中，增量步的步长不可能无限小且增量步位移增量的累积误差可能非常大造成最终结果偏差很大。同时，在位移-载荷曲线的极值点处，切线刚度矩阵为零，位移方程无解。

In theory, if there are no extreme points, that is, each P corresponds to a unique u, then the incremental method can always obtain the true solution. However, in actual engineering applications, the step length of the incremental step cannot be infinitely small, and the cumulative error of the displacement increment in the incremental step may be very large, causing a significant deviation in the final result. At the same time, at the extreme points of the displacement-load curve, the tangent stiffness matrix is zero, and the displacement equation has no solution.

2.1.2 迭代法 2.1.2 Iterative Method

将方程改写成等价的形式

$$\mathrm{u} = \varphi(u)$$

选择一个初始值u_0，将它带入式右端，即可就得

$$u_1 = \varphi(u_0)$$

如此反复迭代，则有

$$u_{k+1} = \varphi(u_k)$$

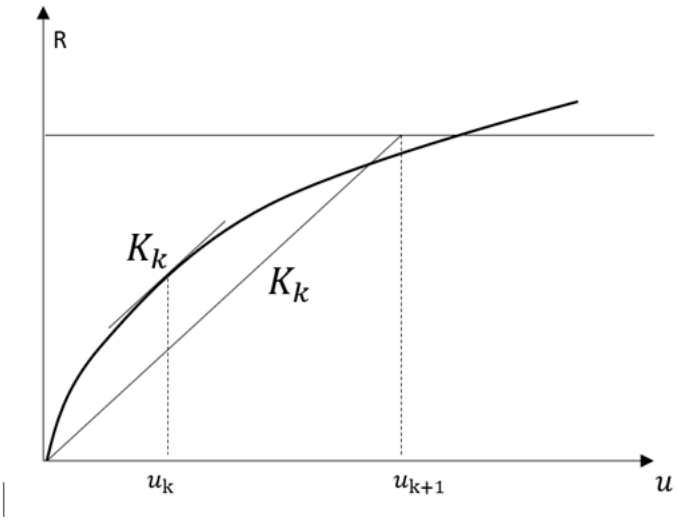
这里，我们称上述方程为迭代方程，如果有$\lim_{\mathrm{k}\to\infty} u_k = u^*$，则称迭代方程收敛，且$u^*$就是非线性方程的解。由此可见，迭代法就是选择合适的迭代方程，用总载荷作用下不平衡的线性解去逼近平衡的线性解，迭代过程就是消除失衡力的过程。

各种迭代法的本质就是迭代方程也就是$\varphi(u)$的取法不同，迭代法主要分为直接迭代法、Newton-Raphson 法、修正 Newton-Raphson 法和 BFGS 法。

2.1.2.1 直接迭代法 2.1.2.1 Direct Iteration Method

直接迭代法取迭代方程为 Direct iteration method takes the iteration equation as

$$u_{k+1} = [K_T(u_k)]^{-1}P$$



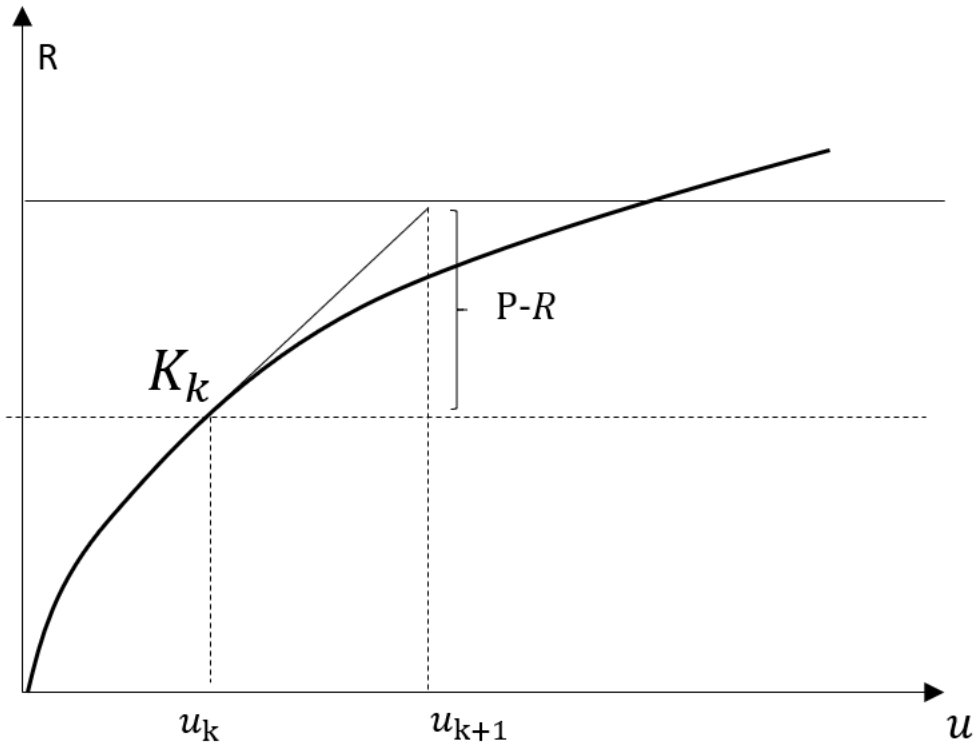
2.1.2.2 Newton-Raphson法 2.1.2.2 Newton-Raphson method

迭代方程 Iterative Equation

$$u_{k+1} = u_k - \left(K_T(u_k)\right)^{-1}\left(P - \mathrm{R}(u_k)\right)$$

详细推导可见附录。K的值和上面的直接法是一样的，但最大区别是第一次迭代的u作为下一次的初始点。

Detailed derivation can be found in the appendix. The value of K is the same as that in the direct method above, but the major difference is that the u from the first iteration is used as the initial point for the next iteration.



2.1.2.3 Modified Newton-Raphson法

2.1.2.3 Modified Newton-Raphson method

如果非线性曲线比较平缓，那么在New-Raphson迭代过程中迭代步的切线刚度矩阵可以近似为上几个或者第一个迭代步的，即

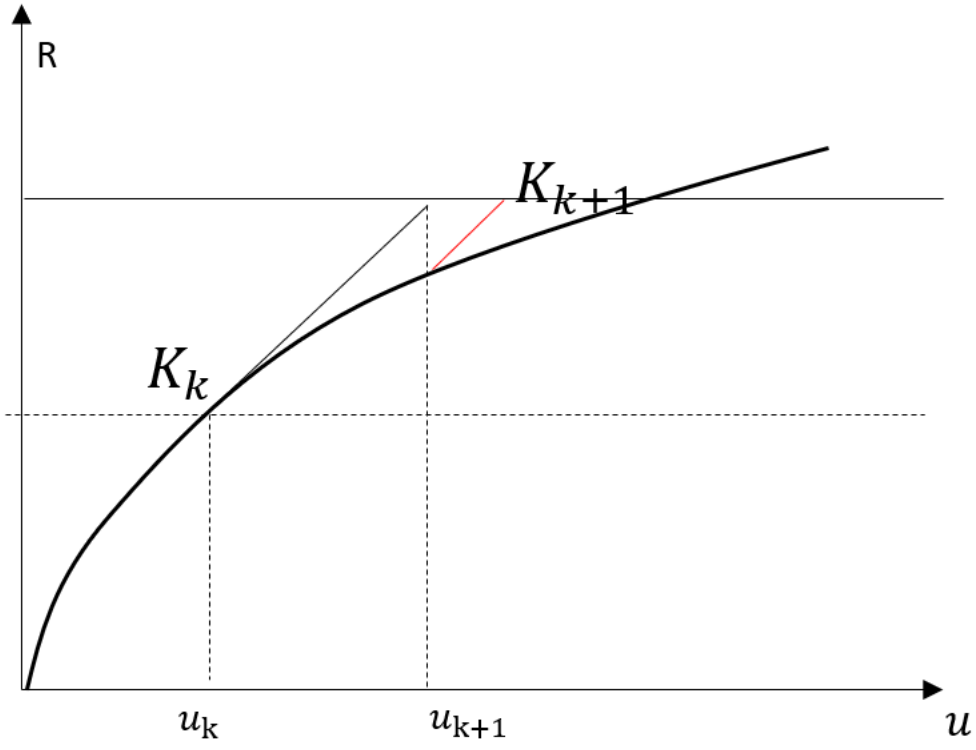
$$\left(K_T(u_{k+1})\right) = \left(K_T(u_k)\right)$$

。这样可以只在第一个迭代步计算一次K，提高每个迭代步的计算速度，但由于一般情况下

If the nonlinear curve is relatively flat, then the tangent stiffness matrix of the iteration step in the New-Raphson iteration process can be approximated as that of the previous few or the first iteration step, i.e.,. This allows for the computation of K only once in the first iteration step, thereby improving the calculation speed of each iteration step. However, due to the generally large changes, this may reduce the convergence speed.

$$K_T(u_k)$$

的变化较大，所以会降低收敛速度。 The changes are relatively large, so it may reduce the convergence speed.



2.1.2.4 BFGS法 2.1.2.4 BFGS method

其实就是Abaqus非线性Step设置中的Quasi-Newton方法，在Abaqus做非线性可以在Newton方法和Quasi-Newton二选一。这里不做更多说明。

It is actually the Quasi-Newton method in the nonlinear Step settings of Abaqus. In Abaqus, nonlinear problems can be solved by choosing either the Newton method or the Quasi-Newton method. No further explanation is provided.

2.1.3 增量迭代法 2.1.3 Incremental Iteration Method

一般情况增量法可以保证求解过程的收敛性但收敛速度较慢，而Newton-Raphson法收敛速度较慢但收敛性没有保证，所以混合法结合了增量法和Newton-Raphson法来求解非线性问题。混合法首先将外载荷分成若干个增量步，在每个增量步内采用Newton-Raphson法迭代求解，在增量步内求解完成后继续求解下一个增量步，最后将所有增量步累加起来即得到结果。

Generally, the incremental method can ensure the convergence of the solution process, but the convergence speed is slow. On the other hand, the Newton-Raphson method has a slow convergence speed but does not guarantee convergence. Therefore, the hybrid method combines the incremental method and the Newton-Raphson method to solve nonlinear problems. The hybrid method first divides the external load into several incremental steps, and uses the Newton-Raphson method for iterative solution within each incremental step. After the solution within the incremental step is completed, the next incremental step is solved, and finally, all the incremental steps are summed up to obtain the result.

2.1.4 收敛判据 2.1.4 Convergence Criteria

常见的收敛判据有分为两种，失衡力准则、位移准则 Common convergence criteria are divided into two types: the imbalance force criterion and the displacement criterion

2.1.4.1 失衡力准则 2.1.4.1 Unbalanced Force Criterion

$$\Delta \mathrm{R}_{max} = \mathrm{P} - \mathrm{R}(u) \leq R_n * \mathrm{R}(u)$$

其中，$\Delta \mathrm{R}_{max}$为最大应力残差，R_n为容许应力残差因子，Abaqus 默认为 5e-3，$\mathrm{R}(u)$为当前应力值。

2.1.4.2 位移准则 2.1.4.2 Displacement Criterion

$$\Delta u^k = u_k - u_{k-1} \leq C_n * u_k$$

其中，Δu^k为第 k 次迭代位移增量，C_n为容许位移误差因子，Abaqus 默认为 1e-2，u_k为第 k 次迭代位移

2.2 Abaqus的非线性问题求解 2.2 Nonlinear Problem Solving in Abaqus

Abaqus的非线性中上面说的三种迭代都涉及，其中General Static分析步采用Newton或者Quasi-Newton方法，而Static, Riks分析步采用Riks弧长法。我们现在只聚焦到General Static的Newton方法。

Abaqus involves all three iterations mentioned above in its nonlinear analysis, where the General Static analysis step uses Newton or Quasi-Newton methods, while the Static and Riks analysis steps use the Riks arc-length method. We will focus on the Newton method used in the General Static analysis now.

详见Abaqus Theory Manual v6.12: 2.2.1 Nonlinear solution methods in Abaqus/Standard

See Abaqus Theory Manual v6.12: 2.2.1 Nonlinear solution methods in Abaqus/Standard

Abaqus/Standard generally uses Newton's method as a numerical technique for solving the nonlinear equilibrium equations. The motivation for this choice is primarily the convergence rate obtained by using Newton's method compared to the convergence rates exhibited by alternate methods (usually modified Newton or quasi-Newton methods) for the types of nonlinear problems most often studied

with Abaqus.

Abaqus/Standard generally employs Newton's method as a numerical technique for solving the nonlinear equilibrium equations. The primary motivation for this choice is the convergence rate obtained by using Newton's method compared to the convergence rates exhibited by alternative methods (usually modified Newton or quasi-Newton methods) for the types of nonlinear problems most commonly studied with Abaqus.

2.2.1 模型例子：非线性弹簧静力分析 2.2.1 Model Example: Nonlinear Spring Static Analysis

Example 1：非线性弹簧例子（详见附件Test30 SpringNonLinear.cae Model-NonLinearSpringY4)

Example 1: Nonlinear Spring Example (see attached Test30 SpringNonLinear.cae Model-NonLinearSpringY4)

本篇采用定义应力应变的关系来模拟非线性的变化过程。

This article uses the relationship between stress and strain to simulate the nonlinear change process.

创建一根线弹簧，左端取固支约束，右端X方向拉力。

Create a linear spring, with a fixed support constraint at the left end and a tensile force in the X direction at the right end.

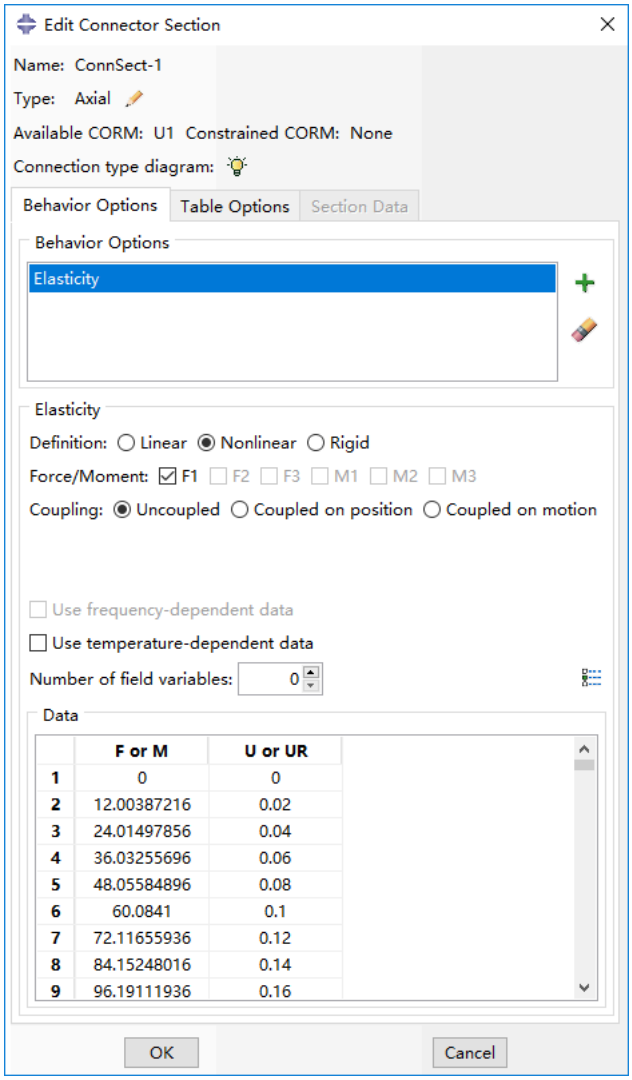
参数如下： Parameters are as follows:

尺寸：X方向长度L=1。 Dimension: Length in X-direction L=1.

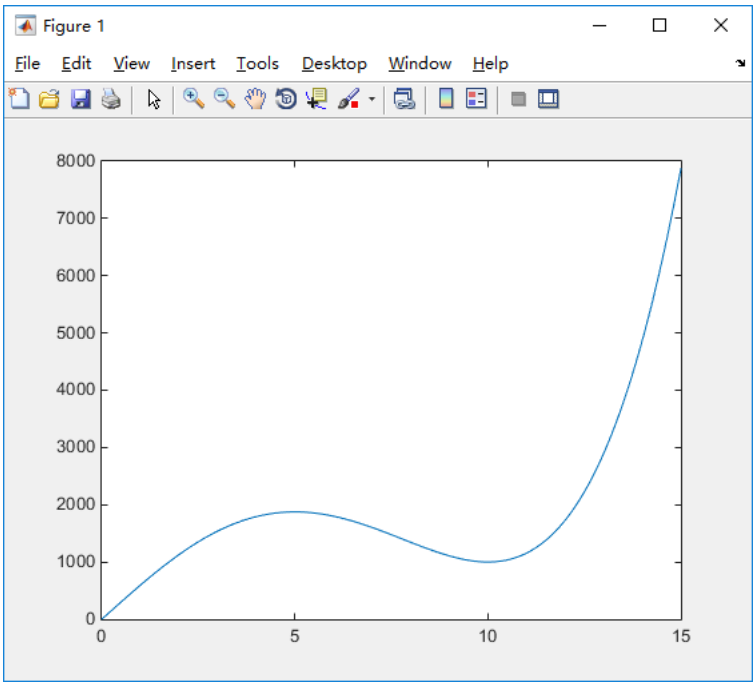
载荷幅值：R=5000 Load Amplitude: R=5000

Abaqus中非线性弹簧设置如下：在Interact模块的Connector Section中，选择类型为Basic，平移为Axis，旋转为None，在Behavior Options中添加Elasticity，定义为Nolinear，在Data表中添加应力应变数据。

The nonlinear spring settings in Abaqus are as follows: In the Connector Section of the Interact module, select the type as Basic, translation as Axis, rotation as None, add Elasticity in Behavior Options, define as Nonlinear, and add stress-strain data in the Data table.

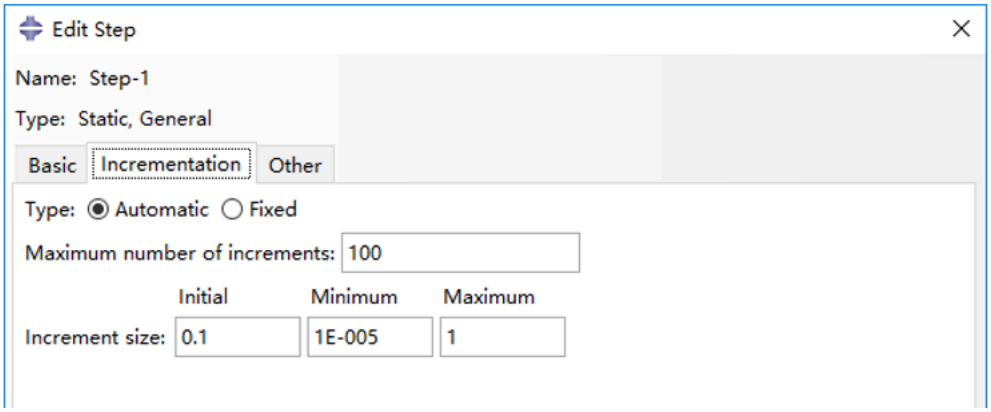


输入的应变-应力曲线如下图所示： The input strain-stress curve is shown in the figure below:



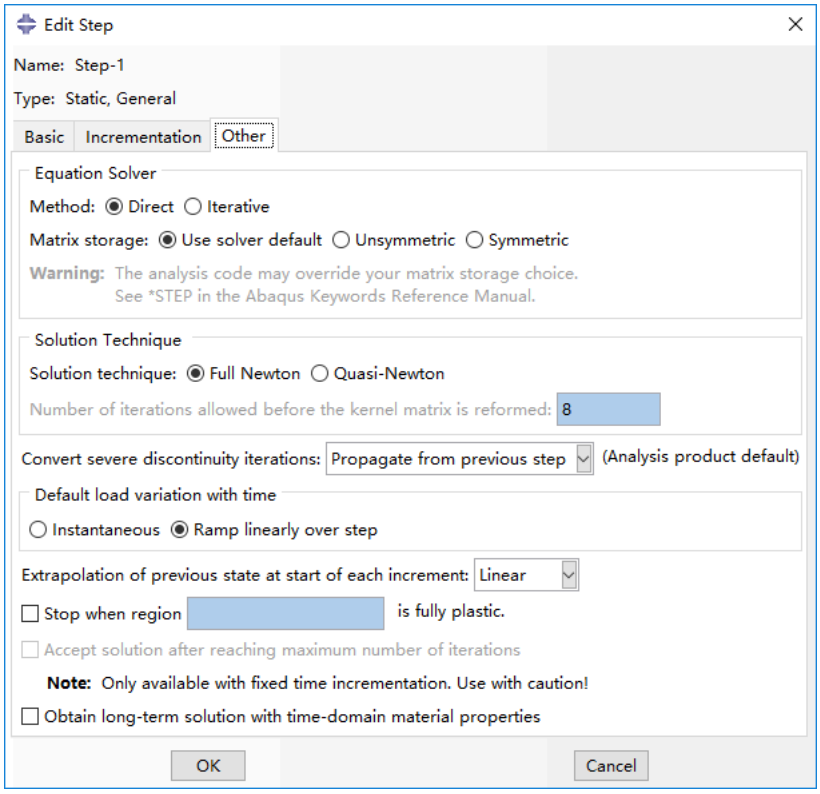
Step选择Static，Generic，Initial Increment Size如果是默认的话就是1，也就是全量，我们这里改为0.1，其他默认。

Step selection: Static, Generic, Initial Increment Size. If the default is used, it is 1, which is the full amount. Here we change it to 0.1, and keep the rest as default.



在Other中默认使用Newton-Raphson方法。

In "Other", the default method is Newton-Raphson.



提交job计算，计算完成后打开对应的*.msg文件和*.sta文件查看我们需要的信息。

Submit job calculation, after the calculation is completed, open the corresponding *.msg file and *.sta file to view the information we need.

Abaqus的Genera Static分析步采用增量迭代法来求解非线性分析问题，默认采用结合增量法和Newton-Raphson法的混合法。

Abaqus's General Static analysis step uses the incremental iterative method to solve nonlinear analysis problems, and the default method is a hybrid method combining the incremental method and the Newton-Raphson method.

下面，笔者将参照这sta文件和msg文件来讲此算例中Abaqus非线性问题的求解步骤。

The author will refer to this sta file and msg file to explain the solution steps of the Abaqus nonlinear problem in this example.

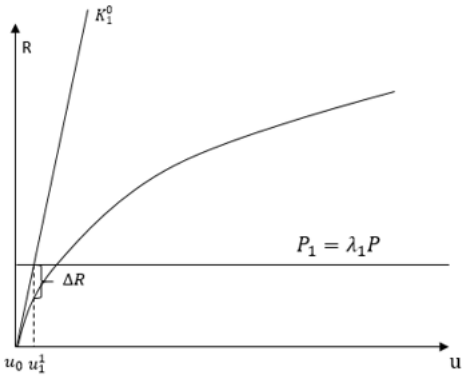
2.2.2 第一增量步 2.2.2 First Increment Step

第一个增量步，取增量步大小$\lambda_1 = \lambda_{initial}$=0.1，$R_1 = \lambda_1 * R = 500$，增量步 i=1。

查看 msg 文件中，可知 Abaqus 在第一个增量步时，增量步大小为 0.1。

```
INCREMENT     1 STARTS. ATTEMPT NUMBER  1, TIME INCREMENT  0.100

NUMBER OF EQUATIONS =        4     NUMBER OF FLOATING PT. OPERATIONS =  3.00E+01

   CHECK POINT   START OF SOLVER

   CHECK POINT  END OF SOLVER

     ELAPSED USER TIME (SEC)       =   0.0000
     ELAPSED SYSTEM TIME (SEC)     =   0.0000
     ELAPSED TOTAL CPU TIME (SEC)  =   0.0000
     ELAPSED WALLCLOCK TIME (SEC)  =          0
```

2.2.2.1 第一迭代步 2.2.2.1 First Iteration Step



迭代步 j=1。

因为是第一个迭代步，所以需要设置初始值：

（1） u_i^{j-1}=$u_{i-1} = u_0 = 0$。其中，下标代表增量步，上标代表迭代步。

（2） 在第一个迭代步时$K_i^{j-1} = K_1^0$为在$u_0 = 0$处的切线刚度，因为是第一个增量步，此时需要重新计算K_1^0的值。

（3） 内力$F(u_i^{j-1}) = F(u_{i-1}) = F(u_0) = 0$。

分析步骤如下：

（1） 得到 K： 根据输入应力应变关系前两个点的数据可得$K_1^0 = \frac{F(0.02)-F(0)}{0.02} = \frac{12.00387216-0}{0.02} = 600.195$

（2） 得到 U：$\Delta R(u_i^{j-1}) = R_i - F(u_i^{j-1}) = R_1 - F(u_0) = 500$,得到$\Delta u^j = \Delta R/K_1^0 = 0.833$，$u_i^j = u_i^{j-1} + \Delta u^j = 0.833$。

（3） 得到更新的ΔR_i^j：根据输入应变应力关系数据在 0.82 和 0.84 线性插值得$F(u_1^1) = 498.077$，$\Delta R(u_i^j) = R_1 - F(u_1^1) = 1.99$，如上图所示。

```
            CONVERGENCE CHECKS FOR EQUILIBRIUM ITERATION     1

AVERAGE FORCE                        187.       TIME AVG. FORCE        187.
LARGEST RESIDUAL FORCE               1.99       AT NODE          2   DOF  1
  INSTANCE:
LARGEST INCREMENT OF DISP.          0.833       AT NODE          2   DOF  1
  INSTANCE: {assembly node}
LARGEST CORRECTION TO DISP.         0.833       AT NODE          2   DOF  1
  INSTANCE: {assembly node}
        FORCE     EQUILIBRIUM NOT ACHIEVED WITHIN TOLERANCE.

 NUMBER OF EQUATIONS =        6     NUMBER OF FLOATING PT. OPERATIONS =  9.10E+01

   CHECK POINT   START OF SOLVER

   CHECK POINT  END OF SOLVER

     ELAPSED USER TIME (SEC)      =   0.0000
     ELAPSED SYSTEM TIME (SEC)    =   0.0000
     ELAPSED TOTAL CPU TIME (SEC) =   0.0000
     ELAPSED WALLCLOCK TIME (SEC) =          0
```

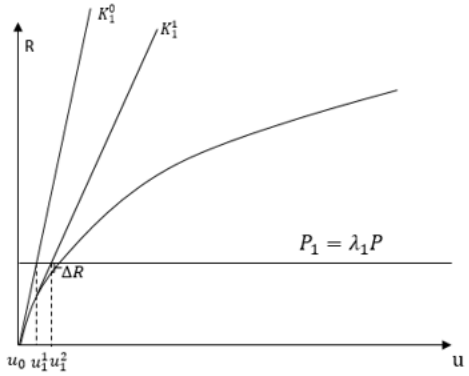
（4） 判据分析：从 Abaqus 的计算结果，我们可以得到在第一个迭代步时，此时，位移增量变化不符合收敛准则，Abaqus 进行第二次迭代。各个物理量的含义如下：

a) LARGEST RESIDUAL FORCE 最大应力残差，即ΔR为 1.99

b) LARGEST INCREMENT OF DISP.最大位移增量，即 dU=$u_1^1 - u_0$为 0.833

c) LARGEST CORRECTION TO DISP.最大修正位移就是不同迭代步对位移增量的修正（在 Abaqus 中，不用将每个迭代步的 dU 记录下来，在计算时只需要增量步的 dU，所以只需要根据当前迭代步对增量步的 dU 进行更新，更新后的结果记录下来放到下一个迭代），即$\Delta u_1 - u_0$为 0.833。结果与我们计算的相同。

2.2.2.2 第二迭代步 2.2.2.2 Second Iteration Step



迭代步 j=2，如图所示，在第二个迭代步时。

分析步骤如下：

（1） 得到 K：$K_i^{j-1} = K_1^1$取应变-应力曲线在u_1^1处的切线刚度$K_1^1 = \frac{F(0.84)-F(0.82)}{0.02} =$ 585.8187。

✧ 注：第二迭代步的 K 取的还是新的刚度，而不是第一个迭代步的 K，由结果和 Abaqus 的一致性，可证明 Abaqus 用的是 Newton 迭代，而不是修正 newton 迭代。

（2） 得到 U：根据K_1^1和ΔR求得$\Delta u^j = \Delta u^2 = 0.034$，$u_i^j = u_i^2 = u_i^{j-1} + \Delta u^j = u_1^1 + \Delta u^2 = 0.8364$，该点恰好还是位于 0.82 和 0.84 之间。

（3） 得到更新的ΔR_i^j：根据输入应变应力关系数据在 0.82 和 0.84 线性插值得$F(u_i^j) = F(u_i^2) = 500$，$\Delta R(u_i^j) = R_1 - F(u_i^2) = 0.00$，注意，此时的外力在一个增量步内都是恒定的。

```
          CONVERGENCE CHECKS FOR EQUILIBRIUM ITERATION     2

AVERAGE FORCE                      188.       TIME AVG. FORCE        188.
LARGEST RESIDUAL FORCE             0.00       AT NODE          2   DOF  1
  INSTANCE:
LARGEST INCREMENT OF DISP.         0.836      AT NODE          2   DOF  1
  INSTANCE: {assembly node}
LARGEST CORRECTION TO DISP.        3.402E-03  AT NODE          2   DOF  1
  INSTANCE: {assembly node}
       THE FORCE     EQUILIBRIUM EQUATIONS HAVE CONVERGED
```

（4） 判据分析：根据 Abaqus 的计算结果，我们可以得到在第一个迭代步时，最大应力残差，即ΔR为 0.00，最大位移增量，即$u_1^2 - u_0$为 0.836，最大修正位移，即$\Delta u^2 - u_0$为 3.402e-03。此时，Abaqus 计算结果满足收敛准则，进入第二个增量步的计算。

✧ 注意：最大位移增量不是这个迭代步内的最大位移$\Delta u^2 = u_1^2 - u_1^1$，而是相对本增量步初始状态的位移增量$\Delta u^2 = u_1^2 - u_0$。

```
ITERATION SUMMARY FOR THE INCREMENT:   2 TOTAL ITERATIONS, OF WHICH
  0 ARE SEVERE DISCONTINUITY ITERATIONS AND  2 ARE EQUILIBRIUM ITERATIONS.

TIME INCREMENT COMPLETED  0.100    ,  FRACTION OF STEP COMPLETED  0.100
STEP TIME COMPLETED       0.100    ,  TOTAL TIME COMPLETED        0.100
```

1.2.3 第二增量步 1.2.3 Second Increment Step

第二个增量步，i=2，取弧长$\lambda_2 = \lambda_1 = 0.1$，$R_i = R_2 = \lambda_2 R + R_1 = 1000$。

1.2.3.1 第一迭代步 1.2.3.1 Step 1.2.3.1

迭代步 j=1。

因为是第一个迭代步，所以需要设置初始值：

（1） $u_i^{j-1} = u_{i-1} = u_1$。也就是上一个增量步的最后一个结果$u_1^2 = 0.8364$。

（2） 在第一个迭代步时$K_i^{j-1} = K_2^0 = K_1$，因为是上一个增量步的 K 在判断收敛后没有计算，此时需要重新计算。（实际上 Abaqus 这边没有计算 K，而是在每次迭代后无论是否收敛都会计算一次 K）

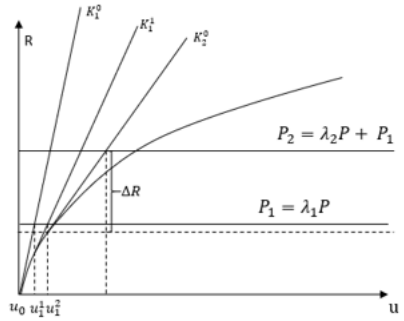
（3） 内力$\mathrm{F}(u_i^{j-1}) = \mathrm{F}(u_{i-1}) = \mathrm{F}(u_1^2) = 500$。

分析步骤如下：

（1） 得到 K：K_2^0为在u_1^2处的切线刚度$K_1^2 = \frac{R(0.84) - R(0.82)}{0.02} = 585.8187$，也就是上一个增量步最后一个迭代步的刚度。

（2） 得到 U：由上面初始值（3），可得$\Delta R(u_i^{j-1}) = R_i - \mathrm{F}(u_i^{j-1}) = R_2 - \mathrm{F}(u_1^2) = 1000 - 500 = 500$，根据$K_2^0$和$\Delta R$求得$\Delta u^j = \Delta u^1 = 0.8535$，$u_i^j = u_2^1 = u_i^{j-1} + \Delta u^j = u_2^0 + \Delta u^1 = 0.8364 + 0.8535 = 1.6899$。

（3） 得到更新的ΔR_i^j：根据输入应变应力关系数据在 1.68 和 1.7 线性插值得$\mathrm{F}(u_2^1) = 973.4694$，$\Delta R(u_i^j) = R_2 - \mathrm{F}(u_2^1) = 26.5306$。



Abaqus 在第二个增量步的结果如下图所示。

```
 INCREMENT     2 STARTS. ATTEMPT NUMBER   1, TIME INCREMENT   0.100

 NUMBER OF EQUATIONS =         6     NUMBER OF FLOATING PT. OPERATIONS =   9.10E+01

     CHECK POINT    START OF SOLVER

     CHECK POINT    END OF SOLVER

        ELAPSED USER TIME (SEC)       =    0.0000
        ELAPSED SYSTEM TIME (SEC)     =    0.0000
        ELAPSED TOTAL CPU TIME (SEC)  =    0.0000
        ELAPSED WALLCLOCK TIME (SEC)  =         0

                CONVERGENCE CHECKS FOR EQUILIBRIUM ITERATION     1


AVERAGE FORCE                          9.998E+02    TIME AVG. FORCE         594.
LARGEST RESIDUAL FORCE                 0.269        AT NODE          2    DOF   1
  INSTANCE:
LARGEST INCREMENT OF DISP.             0.905        AT NODE          2    DOF   1
  INSTANCE: (assembly node)
LARGEST CORRECTION TO DISP.            6.820E-02    AT NODE          2    DOF   1
  INSTANCE: (assembly node)
ESTIMATE OF DISP.     CORRECTION       5.192E-04
FORCE      EQUILIB. ACCEPTED BASED ON SMALL RESIDUAL AND ESTIMATED CORRECTION
          TIME INCREMENT MAY NOW INCREASE TO     0.150

ITERATION SUMMARY FOR THE INCREMENT:    1 TOTAL ITERATIONS, OF WHICH
   0 ARE SEVERE DISCONTINUITY ITERATIONS AND   1 ARE EQUILIBRIUM ITERATIONS.

TIME INCREMENT COMPLETED   0.100     ,   FRACTION OF STEP COMPLETED   0.200
STEP TIME COMPLETED        0.200     ,   TOTAL TIME COMPLETED         0.200
```

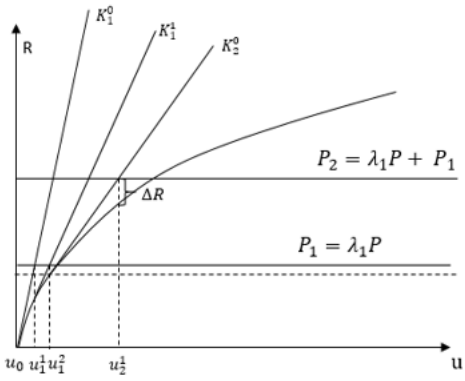
（4） 判据分析：从 Abaqus 的计算结果，我们可以得到在第一个迭代步时，最大应力残差，即ΔR为 0.269，满足应力判断标准 0.269<0.005*9.998e2；

最大位移增量，即$\Delta u^1 = u_2^1 - u_1^2$为 0.905（和理论的$\Delta u^1 = 0.8535$相比，多了 5.15e-2，猜测是每个增量步后 abaqus 都会根据迭代情况做一个线性修正），最大修正位移，即

$c_m = \Delta u^1 - \lambda_2/\lambda_1 * (u_1^2 - u_0) = 6.820e-2$。如果按这个值此时不满足位移收敛条件。

但 Abaqus 还计算了一个 ESTIMAT OF DISP. CORRECTION 5.192e-4 该值是满足位移收敛条件的，而且 FORCE EQUILIB. ACCEPTED BASED ON SMALL RESIDUAL AND ESTIMATED CORRECTION 这句话说了采用了 ESTIMATED CORRECTION 来判据。该值按理论手册多于一个迭代步时才会采用，但不知为何这边也采用了。

此时，Abaqus 计算结果满足收敛准则，进入第三个增量步的计算。



在第二增量步计算结束后，由于第一增量步和第二增量步的迭代次数都小于 5，此时增量步大小增长为原来的 1.5 倍，即 0.15。

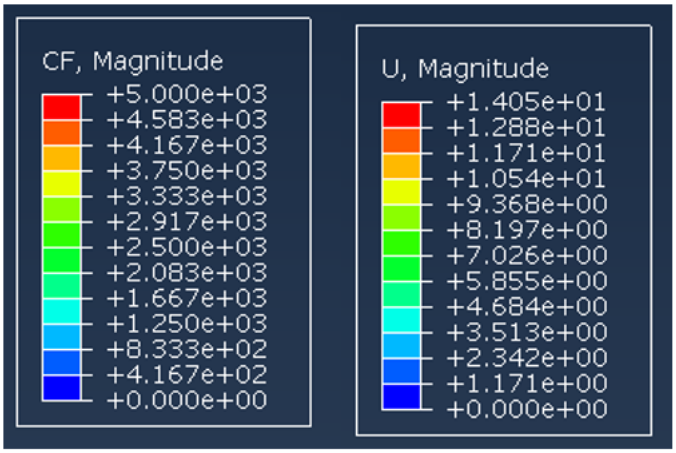
```
FORCE      EQUILIB. ACCEPTED BASED ON SMALL RESIDUAL AND ESTIMATED CORRECTION
          TIME INCREMENT MAY NOW INCREASE TO     0.150
```

2.2.4 第三增量步 2.2.4 Third Increment Step

迭代求解过程同上。 The iterative solution process is the same as above.

2.2.5 最终结果 2.2.5 Final Results

最终应力和应变的计算结果如下图所示。 The calculation results of the final stress and strain are shown in the figure below.



最终计算的增量和每个增量步的迭代次数如下图所示。

The final increments and the number of iterations for each increment step are shown in the figure below.

STEP	INC	ATT	SEVERE DISCON ITERS	EQUIL ITERS	TOTAL ITERS	TOTAL TIME/ FREQ	STEP TIME/LPF	INC OF TIME/LPF	DOF MONITOR	IF RIKS
1	1	1	0	2	2	0.100	0.100	0.1000		
1	2	1	0	1	1	0.200	0.200	0.1000		
1	3	1	0	2	2	0.350	0.350	0.1500		
1	4	1U	0	4	4	0.350	0.350	0.2250		
1	4	2	0	6	6	0.406	0.406	0.05625		
1	5	1U	0	4	4	0.406	0.406	0.05625		
1	5	2	0	4	4	0.420	0.420	0.01406		
1	6	1	0	1	1	0.434	0.434	0.01406		
1	7	1	0	1	1	0.455	0.455	0.02109		
1	8	1	0	1	1	0.487	0.487	0.03164		
1	9	1	0	1	1	0.535	0.535	0.04746		
1	10	1	0	1	1	0.606	0.606	0.07119		
1	11	1	0	1	1	0.713	0.713	0.1068		
1	12	1	0	1	1	0.873	0.873	0.1602		
1	13	1	0	1	1	1.00	1.00	0.1273		

由上图可知，Abaqus在分别在进行第一次计算第四个增量步和第一次计算第五个增量步时，因为求解不收敛，所以增量步大小减为原来的0.25。

From the figure above, it can be seen that when Abaqus is performing the fourth incremental step for the first calculation and the fifth incremental step for the first calculation, the size of the incremental step is reduced to 0.25 of the original due to the non-convergence of the solution.

2.3 总结 2.3 Summary

本文简单介绍了非线性问题的几种常用求解方法，并给出了在结构有限元非线性求解时遵循的收敛准则。同时，给出了一个非线性弹簧静力分析的例子（详见附件SpringNonLinear.cae），并通过这个例子描述了Abaqus采用增

量迭代法求解非线性问题的过程。

This article briefly introduces several commonly used methods for solving nonlinear problems and provides the convergence criteria to be followed in the nonlinear structural finite element analysis. At the same time, an example of a nonlinear spring static analysis is given (see the attachment SpringNonLinear.cae), and through this example, the process of Abaqus using the incremental iterative method to solve nonlinear problems is described.

各种求解方法的优缺点如下： The advantages and disadvantages of various solution methods are as follows:

方法 Method	**优点 Advantages**	**缺点 Disadvantages**
增量法 Incremental method	收敛性较好 Good convergence	收敛效率低，累积误差较大，某些时候会出现增量步过小而无法进行实际计算的问题 Low convergence efficiency, large cumulative error, and sometimes the increment step is too small to perform actual calculations
迭代法 Iterative method	收敛效率较高 High convergence efficiency	收敛性较差 Poor convergence
增量迭代法 Incremental iterative method	收敛性比迭代法好，收敛效率高 Better convergence than iterative methods, with higher convergence efficiency	不能根本上解决迭代法收敛性的问题 The problem of the convergence of iterative methods cannot be fundamentally solved
弧长法 Arc length method	收敛性好，计算稳定性高 Good convergence, high computational stability	增量步过多，求解效率低 Too many incremental steps, low solution efficiency

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

==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