Numerical Computation of Material Tangent in Implicit Finite Element Analysis: Implementation within Abaqus UMAT Subroutine

Aamir Dean, Dr.-Ing. Dr.
dr.aamir.dean@gmail.com
Hannover, Germany

May 12, 2024

Contents

This document offers a brief summary of the numerical computation of the material tangent within implicit Finite Element Analysis (FEA), with a focus on its implementation within the Abaqus UMAT subroutine. It begins with an introduction emphasizing the importance of the material tangent in FEA simulations and the challenges associated with its analytical derivation. The methodology section delves into the use of finite difference methods, particularly the forward difference first-order scheme, for approximating the Jacobian material tangent. Practical considerations such as numerical effects and floating-point arithmetic are discussed in detail. Subsequently, the document outlines a step-by-step methodology for computing the numerical tangent, integrating insights from the newly added section on Finite Difference Methods. Finally, a Fortran code snippet for the UMAT subroutine is provided, illustrating the practical implementation details of the numerical tangent computation. This document serves as a beginner's guide for understanding and implementing the numerical computation of the material tangent in implicit FEA simulations.

1 Introduction

Finite Element Analysis (FEA) is a powerful computational tool widely used in engineering to simulate the behavior of complex structures and materials under various loading conditions. Central to the accuracy and reliability of FEA simulations is the constitutive model that describes the material behavior. This model establishes a fundamental relationship between the applied loads and resulting stresses and strains within the material.

In the context of implicit FEA, the material tangent, also known as the stiffness matrix or the Jacobian (material) tangent/matrix, plays a pivotal role. It represents the sensitivity of material response to changes in strain and is crucial for accurately capturing the nonlinear behavior of materials, particularly in the presence of large deformations, material nonlinearity, and complex loading conditions. The material tangent serves as the cornerstone for solving the system of equations governing the equilibrium of the structure within the implicit finite element framework. It enables the iterative solution process to converge rapidly and accurately by providing information on how the stresses change with respect to small changes in strains. Without an accurate representation of the material tangent, the convergence of the solution may be compromised, leading to inaccurate results and potential divergence of the analysis.

Traditionally, deriving the consistent (algorithmic) Jacobian material tangent, considering nonlinearities, path-dependent effects, and other complexities in the material behavior, can pose significant challenges, especially with complex material models or non-linear behaviors. Analytically deriving this tangent can be time-consuming and demanding. Moreover, any modifications to the constitutive equations require corresponding adjustments in the tangent operator. The consistent tangent is essential for ensuring the numerical stability and accuracy of FEA simulations. It provides a reliable approximation of the material response to changes in strain, accounting for nonlinear effects and ensuring consistency across iterations. This consistency is crucial for maintaining the convergence of iterative solution processes and obtaining accurate predictions of structural behavior and performance.

To streamline the development of constitutive models and circumvent the laborious analytical derivation process, numerical methods for computing (approximating) the algorithmic consistent material tangent offer a practical workaround. Numerically computing the algorithmic consistent material tangent involves approximating the tangent operator using finite difference methods or automatic differentiation. However, it's important to be aware of potential numerical effects, such as cancellation, round-off errors, truncation errors, step size selection, and instability, when utilizing numerical methods to compute the algorithmic consistent material tangent, especially considering the problem of cancellation due to floating-point arithmetic, see [1].

2 Numerical Computation: Finite Difference Methods

Finite difference methods are numerical techniques used to approximate derivatives of functions by discretizing the derivative formula over a finite interval. There are several types of finite difference methods, including forward, backward, and central differences. These methods offer a straightforward way to compute derivatives numerically, but they come with limitations such as truncation errors and sensitivity to the choice of step size h. Careful selection of the step size is crucial to balance accuracy and stability. Additionally, floating-point arithmetic can introduce cancellation errors, particularly when subtracting two nearly equal numbers. Despite these challenges, finite difference methods provide a practical workaround for computing the algorithmic consistent material tangent.

Herein, the numerical computation is explained using a forward difference first-order scheme (characterized by its simplicity and ease of implementation). In a forward difference first-order scheme, the derivative of a function $f(x)$ at a point x_i is approximated using the function values at x_i and x_{i+1}. Mathematically, it can be expressed as:

$$f'(x) \approx \frac{f(x_{i+1}) - f(x_i)}{h},$$

where is $f'(x)$ the approximation of the derivative of $f(x)$ at x_i, $f(x_i)$ and $f(x_{i+1})$ are the function values at x_i and x_{i+1}, respectively, and h is the step size or spacing between the points x_i and x_{i+1} (also known as perturbation parameter). This method provides an estimate of the slope of the function over a small interval $[x_i, x_{i+1}]$ and is first-order accurate, meaning that the error in the approximation decreases linearly with the step size h.

The quality of the derivative approximation critically depends on the chosen step size h. As h approaches zero ($h \to 0$) the approximation tends toward the exact derivative $f'(x)$. However, in practical computation, the value of h is constrained by the limitations of floating-point arithmetic. Therefore, limits on the step size h are introduced, taking into account the machine precision ϵ_m:

$$h \geq \sqrt{\epsilon_m} \text{ and } h \gg \mathrm{Abs}\left(x_i \cdot \epsilon_m\right).$$

These limits ensure that the step size h remains sufficiently large to avoid numerical instability while still providing an accurate approximation of the derivative. For detailed derivations of these limits, please refer to [1, 3]. The machine precision ϵ_m, indeed depends on the type of machine and the floating-point arithmetic used. For single-precision machines, where floating-point numbers are typically represented using 32 bits, ϵ_m is approximately 1.19×10^{-7}. For double-precision machines, which use 64 bits to represent floating-point numbers, ϵ_m is approximately 2.22×10^{-16}. These values represent the smallest positive floating-point number that can be added to 1 to obtain a value different from 1.

It's important to note that the machine precision can vary depending on the hardware architecture, compiler settings, and implementation of floating-point arithmetic. Therefore, when performing numerical computations, it's advisable to use the appropriate machine precision for the specific hardware and arithmetic environment to ensure accurate and reliable results.

3 Methodology: Computing the Numerical Tangent

In this section, we outline the methodology for computing the numerical tangent within the context of implicit Finite Element Analysis (FEA), focusing on its implementation within the Abaqus UMAT subroutine.

The constitutive law establishes a crucial relationship between the applied strain $\boldsymbol{\varepsilon}$ (second-order tensor referred to as "agency") and the resulting stress $\boldsymbol{\sigma}$ (second-order tensor termed as "response"). In Abaqus, this linkage can be realized through the implementation of a material routine within the UMAT user interface. The applied strain $\boldsymbol{\varepsilon}$ is characterized by the one-dimensional fields STRAN(6) (total strain) and DSTRAN(6) (strain increments), which capture the deformation of the material. Subsequently, the UMAT routine computes the corresponding stresses response $\boldsymbol{\sigma}$, which needs to be communicated back via the one-dimensional field variable STRESS(6). Additionally, the Jacobian material tangent $\mathcal{C}$ (fourth-order tensor), representing the partial derivatives of the stress with respect to strain components, is conveyed through the two-dimensional field variable DDSDDE(6×6).

To utilize the forward difference first-order scheme for computing the Jacobian, the following procedure is employed: Each of the six components of the strain vector STRAN(6) is individually perturbed by adding a disturbance value ε_d. In each perturbation step j (where j ranges from 1 to 6) only one component of the strain vector STRAN(6) is disturbed, resulting in a perturbed strain vector STRAN$_d$(6), while leaving the remaining strain components undisturbed. For each perturbation step j, a column in the matrix of disturbed stresses is obtained, denoted as STRESS$_d$(6×6), where the second index indicates the perturbation step j and the first index i (where i ranges from 1 to 6) represents the stress response component of that step. The undisturbed stress vector (current stress vector) $\text{STRESS}_{ud}(6)$ is obtained during an undisturbed step ($i = 7$), that is doing the computation of the material routine with the undisturbed strain vector (current strain vector) $\text{STRAN}_{ud}(6)$.

By observing the stress variations induced by these perturbations, the partial derivatives of stress with respect to each strain component, and hence the Jacobian material tangent $\mathcal{C}$ can be approximated:

$$\mathfrak{C}\,(i,j) = \frac{\text{STRESS}_d(i,j) - \text{STRESS}_{ud}(i)}{\boldsymbol{\varepsilon}_d\,(j)},$$

where $\mathfrak{C}(6 \times 6)$ is the matrix representation of the approximated fourth-order Jacobian material tangent tensor $\mathcal{C}$ and $\boldsymbol{\varepsilon}_d\,(6)$ is the perturbation strain vector. The Jacobian matrix $\mathfrak{C}(6 \times 6)$ is passed back via the field DDSDDE(6×6) in ABAQUS UMAT. The value of the perturbation strain ε_d is limited due to the floating-point representation, as explained above. In case a double-precision environment is used, the limits for the perturbation strain ε_d for each perturbation step j is given by:

$$\boldsymbol{\varepsilon}_d\,(j) = \max\left(1.0 \times 10^{-8}, \text{Abs}\left(1.0 \times 10^{-8} \cdot \text{STRAIN}_{ud}(j)\right)\right). \tag{1}$$

4 Implementation: Fortran Code for Abaqus UMAT Subroutine

Presented below is a Fortran code snippet for the UMAT subroutine, tailored to compute the numerical Jacobian material tangent. For additional details regarding the UMAT subroutine, please refer to [2]. The code is also available on GitHub[1]

Listing 1: UMAT with Numerical Jacobian Material Tangent Computation

```
      SUBROUTINE UMAT(STRESS, STATEV, DDSDDE, STRAN, DSTRAN, TIME,
     #   DTIME, TEMP, DTEMP, PREDEF, DPRED, CMNAME, NDI, NSHR, NTENS,
     #   NSTATV, PROPS, NPROPS, COORDS, DROT, PNEWDT, CELENT, DFGRD0,
     #   DFGRD1, NOEL, NPT, LAYER, KSPT, KSTEP, KINC)

      IMPLICIT REAL*8(A-H,O-Z)

      DIMENSION STRESS(NTENS), STATEV(NSTATV), DDSDDE(NTENS,NTENS)
      DIMENSION STRAN(NTENS), DSTRAN(NTENS), TEMP(1), DTEMP(1)
      DIMENSION PREDEF(1), DPRED(1), PROPS(NPROPS), COORDS(3)
      DIMENSION DROT(3,3), DFGRD0(3,3), DFGRD1(3,3)

      ! PARAMETERS
      INTEGER, PARAMETER :: NUM_STEPS = 6
      REAL*8, PARAMETER :: MACHINE_EPSILON = 1.0D-8

      ! LOCAL VARIABLES
      REAL*8 :: STRESS_UD(NTENS), STRESS_D(NTENS, NUM_STEPS)
      REAL*8 :: STRAN_UD(NTENS), STRAN_D(NTENS, NUM_STEPS)
      REAL*8 :: EPSILON_D

      ! INITIALIZE THE UNDISTURBED STRAIN VECTOR
      STRAN_UD = STRAN

      ! LOOP OVER EACH PERTURBATION STEP
      DO I = 1, NUM_STEPS

      ! PERTURB ONLY ONE COMPONENT OF THE STRAIN VECTOR AT A TIME
```

[1]https://github.com/aamir-dean/Numerical-Tangent-UMAT

```
      STRAN_D = STRAN_UD
      ! CALCULATE THE PERTURBATION VALUE BASED ON MACHINE EPSILON AND
         CURRENT STRAIN COMPONENT
      EPSILON_D = MAX(MACHINE_EPSILON,
   #  ABS(MACHINE_EPSILON * STRAN_UD(I)))
      STRAN_D(I) = STRAN_UD(I) + EPSILON_D

      ! COMPUTE THE DISTURBED STRESS MATRIX FOR THE PERTURBED STRAIN
      CALL MATERIAL_ROUTINE(STRESS_D(:, I), STATEV, STRAN_D,
   #  DSTRAN, TIME, DTIME, TEMP, DTEMP, PREDEF, DPRED, CMNAME,
   #  NDI, NSHR, NTENS, NSTATV, PROPS, NPROPS, COORDS, DROT,
   #  PNEWDT, CELENT, DFGRD0, DFGRD1, NOEL, NPT, LAYER, KSPT,
   #  KSTEP, KINC)

      ENDDO

      ! COMPUTE THE UNDISTURBED STRESS VECTOR AND UPDATED HISTORY VARIABLES
      CALL MATERIAL_ROUTINE(STRESS_UD, STATEV, STRAN_UD,
   #  DSTRAN, TIME, DTIME, TEMP, DTEMP, PREDEF, DPRED, CMNAME,
   #  NDI, NSHR, NTENS, NSTATV, PROPS, NPROPS, COORDS, DROT,
   #  PNEWDT, CELENT, DFGRD0, DFGRD1, NOEL, NPT, LAYER, KSPT,
   #  KSTEP, KINC)

      ! COMPUTE THE APPROXIMATED CONSISTENT JACOBIAN MATERIAL TANGENT
     DO I = 1, NTENS
     DO J = 1, NUM_STEPS
      !CALCULATE THE PERTURBATION VALUE BASED ON MACHINE EPSILON AND
         CURRENT UNDISTURBED STRAIN COMPONENT
      EPSILON_D = MAX(MACHINE_EPSILON,
   #  ABS(MACHINE_EPSILON * STRAN_UD(J)))
      !COMPUTE THE FINITE DIFFERENCE APPROXIMATION OF THE TANGENT
      DDSDDE(I, J) = (STRESS_D(I, J) -
   #  STRESS_UD(I)) / EPSILON_D
      ENDDO
      ENDDO

      ! ASSIGN THE FINAL STRESS VECTOR
      STRESS = STRESS_UD

      RETURN
      END SUBROUTINE UMAT
```

References

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007.

[2] M. Smith. *ABAQUS/Standard User's Manual, Version 6.9*. Dassault Systemes Simulia Corp, United States, 2009.

[3] M. Vogler. *Anisotropic Material Models for Fiber Reinforced Polymers*. Mitteilungen des Instituts für Statik und Dynamik der Leibniz-Universität Hannover. ISD, 2015.