

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 10, 2024

Contents

1 Introduction	2
2 Methodology: Computing the Numerical Tangent	3
3 Implementation: Fortran Code for Abaqus UMAT Subroutine	3

This document provides a guide to the numerical computation of the material tangent within the context of implicit Finite Element Analysis (FEA), focusing on its implementation within the ABAQUS UMAT subroutine.

The document begins with an introduction that outlines the importance of the material tangent in FEA and the challenges associated with its analytical derivation. It discusses the rationale behind utilizing numerical methods to compute the tangent and highlights considerations such as numerical effects and floating-point arithmetic.

Subsequently, the methodology section delves into the process of computing the numerical tangent using finite difference methods, with a particular emphasis on the forward difference first-order scheme. It explains the step-by-step procedure for perturbing the strain components and computing the corresponding stress responses to approximate the Jacobian material tangent.

Furthermore, the document presents a Fortran code snippet for the UMAT subroutine, tailored to implement the numerical tangent computation. The code demonstrates the practical implementation details, including perturbing the strain components, computing stresses, and approximating the material tangent using finite differences.

1 Introduction

For implicit Finite Element (FE) analysis, the Jacobian material tangent is essential. However, deriving the consistent (algorithmic) Jacobian material tangent, considering non-linearities, 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.

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].

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 $f'(x)$ is 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 ($h \rightarrow 0$) 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 :

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

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.

2 Methodology: Computing the Numerical Tangent

The constitutive law establishes a crucial relationship between the applied strain $\boldsymbol{\epsilon}$ (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 $\boldsymbol{\epsilon}$ 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 DDSDDDE(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) 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) 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:

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

where $\mathfrak{C}(6 \times 6)$ is the matrix representation of the approximated fourth-order Jacobian material tangent tensor \mathcal{C} and $\epsilon_d(6)$ is the perturbation strain vector. The Jacobian matrix $\mathfrak{C}(6 \times 6)$ is passed back via the field DDSDDDE(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:

$$\epsilon_d(j) = \max(1.0 \times 10^{-8}, \text{Abs}(1.0 \times 10^{-8} \cdot \text{STRAIN}_{ud}(j))). \quad (1)$$

3 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]

Listing 1: UMAT with Numerical Jacobian Material Tangent Computation

```

1  SUBROUTINE UMAT(STRESS, STATEV, DDSDE, STRAN, DSTRAN, TIME,
2  # DTIME, TEMP, DTEMP, PREDEF, DPRED, CMNAME, NDI, NSHR, NTENS,
3  # NSTATV, PROPS, NPROPS, COORDS, DROT, PNEWDT, CELENT, DFGRD0,
4  # DFGRD1, NOEL, NPT, LAYER, KSPT, KSTEP, KINC)
5
6  IMPLICIT REAL*8(A-H,O-Z)
7
8  DIMENSION STRESS(NTENS), STATEV(NSTATV), DDSDE(NTENS,NTENS)
9  DIMENSION STRAN(NTENS), DSTRAN(NTENS), TEMP(1), DTEMP(1)
10 DIMENSION PREDEF(1), DPRED(1), PROPS(NPROPS), COORDS(3)
11 DIMENSION DROT(3,3), DFGRD0(3,3), DFGRD1(3,3)
12
13 ! PARAMETERS
14 INTEGER, PARAMETER :: NUM_STEPS = 6
15 REAL*8, PARAMETER :: MACHINE_EPSILON = 1.0D-8
16
17 ! LOCAL VARIABLES
18 REAL*8 :: STRESS_UD(NTENS), STRESS_D(NTENS, NUM_STEPS)
19 REAL*8 :: STRAN_UD(NTENS), STRAN_D(NTENS, NUM_STEPS)
20 REAL*8 :: EPSILON_D
21
22 ! INITIALIZE THE UNDISTURBED STRAIN VECTOR
23 STRAN_UD = STRAN
24
25 ! LOOP OVER EACH PERTURBATION STEP
26 DO I = 1, NUM_STEPS
27
28 ! PERTURB ONLY ONE COMPONENT OF THE STRAIN VECTOR AT A TIME
29 STRAN_D = STRAN_UD
30 ! CALCULATE THE PERTURBATION VALUE BASED ON MACHINE EPSILON AND
    CURRENT STRAIN COMPONENT
31 EPSILON_D = MAX(MACHINE_EPSILON,
32 # ABS(MACHINE_EPSILON * STRAN_UD(I)))
33 STRAN_D(I) = STRAN_UD(I) + EPSILON_D
34
35 ! COMPUTE THE DISTURBED STRESS MATRIX FOR THE PERTURBED STRAIN
36 CALL MATERIAL_ROUTINE(STRESS_D(:, I), STATEV, STRAN_D,
37 # DSTRAN, TIME, DTIME, TEMP, DTEMP, PREDEF, DPRED, CMNAME,
38 # NDI, NSHR, NTENS, NSTATV, PROPS, NPROPS, COORDS, DROT,
39 # PNEWDT, CELENT, DFGRD0, DFGRD1, NOEL, NPT, LAYER, KSPT,
40 # KSTEP, KINC)
41
42 ENDDO
43
44 ! COMPUTE THE UNDISTURBED STRESS VECTOR AND UPDATED HISTORY VARIABLES
45 CALL MATERIAL_ROUTINE(STRESS_UD, STATEV, STRAN_UD,
46 # DSTRAN, TIME, DTIME, TEMP, DTEMP, PREDEF, DPRED, CMNAME,
47 # NDI, NSHR, NTENS, NSTATV, PROPS, NPROPS, COORDS, DROT,
48 # PNEWDT, CELENT, DFGRD0, DFGRD1, NOEL, NPT, LAYER, KSPT,
49 # KSTEP, KINC)
50
51 ! COMPUTE THE APPROXIMATED CONSISTENT JACOBIAN MATERIAL TANGENT
52 DO I = 1, NTENS
53 DO J = 1, NUM_STEPS
54 !CALCULATE THE PERTURBATION VALUE BASED ON MACHINE EPSILON AND

```

```
55      CURRENT UNDISTURBED STRAIN COMPONENT
56      EPSILON_D = MAX(MACHINE_EPSILON ,
57      # ABS(MACHINE_EPSILON * STRAN_UD(J)))
58      ! COMPUTE THE FINITE DIFFERENCE APPROXIMATION OF THE TANGENT
59      DDSDE(I, J) = (STRESS_D(I, J) -
60      # STRESS_UD(I)) / EPSILON_D
61      ENDDO
62      ENDDO
63      ! ASSIGN THE FINAL STRESS VECTOR
64      STRESS = STRESS_UD
65
66      RETURN
67      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.