A comparison of different methods for calculating tangent-stiffness matrices in a massively parallel computational peridynamics code.

Michael D. Brothers, John T. Foster*, Harry R. Millwater*

Mechanical Engineering Department, The University of Texas at San Antonio

Abstract

matrix calculation methods including a newly developed Complex Taylor Series Expansion (CTSE)-based "complex-step" method alongside established methods: forward-difference, central-difference and automatic differentiation. Tangent-stiffness matrix is a term from computational mechanics which here refers to the Jacobian matrix as used in first-order Newton-Raphson methods for the solution of non-linear algebraic systems of equations developed to describe physical or theoretical systems studied in math, science, engineering and business. To perform the comparative study, the above tangent-stiffness calculation methods were applied in a massively parallel computational peridynamics code developed at Sandia National Labs, called *Peridigm*. In the comparative study, for each datum for each run, complex-step was multiple orders of magnitude more accurate the finite-difference methods, according to a common comparison made to the automatic-differentiation method datum. However, for the implementation contemporary to the study, complex-step was also the slowest method among the four for each datum for each run. A mathematical definition of *Michael:* ~~the methods~~complex-step and and automatic differentiation, description of the computer implementations and justification of the comparative study precede results for clarity. The intended audience of this paper includes researchers, professionals and students, *Michael:* ~~therefore conceptually explicit language appealing to multiple levels of understanding is used~~therefor readers are directed to selected background sources in the text for well-known concepts while less familiar concepts are derived

*Corresponding Author: john.foster@utsa.edu

Preprint submitted to Computer Methods in Applied Mechanics and Engineering *August 28, 2013*

in-paper with an attempt made to lay steps out plainly for those in disparate fields or who are new to their studies.

Keywords: Newton's method, Newton-Raphson, numeric differentiation, complex-step, finite-difference, automatic-differentiation, Jacobian, tangent-stiffness
2010 MSC: 65D25

1. Acknowledgements

This work was funded in part by a grant from the United States Airforce Office of Scientific Research. This work is affiliated with the Center for Simulation Visualization and Real-Time Prediction at the University of Texas at San Antonio.

2. Introduction

In order to maintain the quadratic convergence properties of the first-order Newton's method in quasi-static nonlinear analysis of solid structures it is crucial to obtain accurate, algorithmically consistent tangent-stiffness matrices. For an extremely small class of nonlinear material models, these consistent tangent-stiffness operators can be derived analytically; however, most often in practice, they are found through numerical approximation of derivatives.

A goal of this comparative study, was to develop and evaluate a new, accurate, and practical method for calculating tangent-stiffness matrices against established methods. This new method is based on a complex number Taylor series expansion and refered to as CTSE or the "complex-step" method. The distinction of 'accurate' is defined by comparison of the new method to popular finite differencing techniques used for computing tangent-stiffness matrices and with the exact algorithmically consistent derivatives computed with *automatic differentiation*.

It was thought that in addition to comparative information regarding the new method, comparisons of exclusively finite-difference, central-difference and automatic differentiation could prove valuable for developers or analysts not necessarily considering complex-step but needing data to help decide whether or not to invest in automatic-differentiation vs. finite-difference. A comparison and discrimination of the methods was to be achieved through application and the measurement of results in-situ. The results would largely be treated as deterministic consequences of method choice. There was not an attempt made to mathematically prove the suitability of one method over another of the methods. The study was retrospective, as explained in section SECTION of the instant paper.

The scope of the application component of the study was limited to the Linear Peridynamic Solid material model implemented in the computational peridynamics code, *Peridigm*. Identifying a specific application provided a practical framework for implementing the new method and solving engineering problems to generate the data needed to compare the methods. In particular, *Peridigm* was chosen cause it combined several helpful characteristics: The reliance on Newton's method and tangent-stiffness matrices for solving quasi-static implicit problems, prior inclusion of finite-difference, central-difference and automatic-differentiation based methods needed for comparison to a new method that would be developed, and being largely based in the Ansi standard version of the popular C++ programming language.

The aim of this paper is to introduce new information, but also to compile prior art to serve as a general reference for solving non-linear systems. *Michael:* ~~The former is mainly intended for an audience of experienced researchers and professionals who are seeking novel tools, the later is intended for an audience of undergraduate to graduate level students who would benefit from a source detailing basic methodology. According to this aim, concepts are presented in multiple levels of detail to accommodate both a broad and deep audience.~~ After giving background on the *Michael:* possibly less familiar differentiation techniques underlying the methods to be discussed here, presented in this paper is: a description of tangent-stiffness matrices and how to use them to solve non-linear systems, detailed directions for producing tangent-stiffness matrices with each the methods identified above *Michael:* ~~preceded by background as it becomes necessary~~, a description and justification of a new method, called 'complex-step' for calculating tangent-stiffness matrices, a description of implementing complex-step in the software package used in the comparative study, a description of the parameters of the comparative study conducted to rank the methods, a presentation and analysis of the results of the comparative study, and conclusions and thoughts on potential future work. Finally, the reader is referred to the corresponding author's website for c++ source-code and data necessary to reproduce the results presented here. Additionally included on the website is a C+ template library with classes for solving non-linear systems using the techniques dicussed here, as well as two validation and verification example problemsi which show how to use the library.

2.1. Differentiation Techniques

This subsection contains background information on the differentiation techniques underlying the tangent-stiffness matrix calculation methods compared in the study. Since first-order finite difference techniques are considered well known, they are not described where. Instead the reader is referred to [1, Chap. 4.1.3].

2.1.1. Basis of New Method, 'Complex Step', in CTSE

Michael: [TODO: cite lyness and moler, complete section]

During a literature review in preparation of material distinct from the instant paper, one of the instant authors identified a work presenting an application of the prior work of applied mathematicians Lyness and Moler to the numerical approximations of the first derivatives of real functions. First called Complex Taylor Series Expansion (CTSE) in (Lyness and Moler) and later by [2], where Dr. Millwater is one of the instant authors, the use of the term and its acronym are adopted out of deference to their originators. Lyness and Moler described CTSE based on approximating the Taylor Series Expansion of a function F, where $F : R^1 \to R^1$ about a complex point and solving for the first derivative. The application of Lyness and Moler's work demonstrated by Squire and Trap showed that there was empirical evidence that a CTSE based numerical derivative formula could resist the accuracy reducing phenomenon of subtractive cancellation in circumstances where central-difference and forward-difference were demonstrated to succumb to subtractive-cancellation.

Michael:

~~The types of first-order finite-difference that will be discussed are forward-difference and central difference. CTSE and these finite-difference techniques are similar in that they have been developed by algebraically solving approximated Taylor series expansions for first derivative terms.~~

2.1.3. Differentiation Technique: Automatic-differentiation

Automatic-differentiation or 'AD' is a computerized method for computing exact derivatives based the chain-rule from calculus. AD takes advantage of the fact that any mathematical function executed on a computer, no matter how complicated, is a "composition of simple operations" (add, multiply, power, transcendental and the like) each having known analytical derivatives

[3]. For reference, the AD implementation used in the study is the "Sacado" package from the "Trilinos" library developed out of Sandia National Labs [4].

The way an AD system works is by first evaluating the innermost function of the composition, then presenting that function's output as input to the next level function until all levels are complete, in a way no different from how a normal computer program or human would evaluate composition functions. AD departs in that as each nested function is evaluated, the function's partial derivative with respect to the designated variables of the given input is also calculated. This is possible because the elementary math functions are hard-coded into the AD source-code along with their analytical derivatives, and linked by special instructions, so that when the elementary math functions are called upon for computation, their partial derivatives may be computed and stored in a sequence. The AD system then multiples the final sequence of partial derivatives together to produce the exact equivalent to taking a partial derivative of the corresponding composition function made up of the elementary functions with respect to a designated variable at a particular value. It is obviously, but bears mentioning, that the AD system could simply store one value for partial derivatives, modifying it as appropriate for every function evaluation rather than keeping a sequence. In the literature, the AD scheme described here is called 'forward automatic-differentiation'. For brevity, only forward AD will be covered since it is pertinent to the study, however the reader is referred to the introduction section of [5] and its references list for further information on AD, particularly [6] which is foundational.

To make the above description complete, this subsection concludes with an example forward AD process, with an emphasis on how general purpose information known to the computer is specialized with additional information from the problem and used to carry out AD:

1. Take an example composition function $f(x) = (sin(cos(x)))^2$

2. Suppose we want to know $\frac{df}{dx} \mid_{x_0}$ where x_0 is a particular value of x.

    (a) Given that:

        i. $S \in R^1$
        ii. $X \in S$

iii. $g, h, k \in H : S \to S$
iv. $\forall s \in S : g(s) = s^2,\ h(s) = sin(s),\ k(s) = cos(s)$
v. $\forall x \in X : f(x) = g(h(k(x)))$

(b) Given that the computer is programmed with some mathematical definitions:

i. A. $u, v, w \in H : R^3 \to R^1$

B. $s, a, b, x \in R^1$

ii. particular values can be identified:
$s = s_0, .., s_n, ...s_\infty \mid n = [0, \infty)$
and similarly for the other variables a, b, x

iii. functions u, v, w and their partial derivatives w.r.t s are defined such that:

for a, b, s equal to a_n, b_n, s_n:

$u \mid_{a_n, b_n, s_n} = a_n \cdot s_n^{b_n}$	$\frac{\partial u}{\partial s} \mid_{a_n, b_n, s_n} = a_n \cdot b_n \cdot s_n^{b_n - 1}$
$v \mid_{a_n, b_n, s_n} = a_n \cdot sin(b_n \cdot s_n)$	$\frac{\partial v}{\partial s} \mid_{a_n, b_n, s_n} = a_n \cdot b_n \cdot cos(b_n \cdot s_n)$
$w \mid_{a_n, b_n, s_n} = a_n \cdot cos(b_n \cdot s_n)$	$\frac{\partial w}{\partial s} \mid_{a_n, b_n, s_n} = -a_n \cdot b_n \cdot sin(b_n \cdot s_n)$

(c) Given that it is possible to describe $f(x) = (sin(cos(x)))^2$ in terms the computer understands by inputting $f(x)$ such that the computer stores an equivalent statement $f(x) = u(a, b, s) \mid_{arguments}$, iff the arguments of u, v, w are chosen such that u, v, w approximate g, h, k as follows:

Function	a	b	s	Approximates Function
u	1	2	v	g
v	1	1	w	h
w	1	1	x	k

3. It follows from 2(a)v that we can evaluate $\frac{d}{dx} f(x) \mid_{x_0}$ with the chain rule:

$\frac{d}{dx} f(x) = \frac{d}{dx} \cdot g(h(k(x))) \mid_{x_0}$
$\frac{d}{dx} f(x) = \frac{dg}{dh} \cdot \frac{dh}{dk} \cdot \frac{dk}{dx} \mid_{x_0}$

From the rest of 2a it follows that we can approximate $\frac{d}{dx} f(x) \mid_{x_0}$ by specializing the computer's general forms of u, v, w according to 2c, with parameters a, b chosen for each function and held as constant, and evaluating, such that the total derivative our original function w.r.t x where $x = x_0$ is approximated by the partial derivative of our equivalent statement, $f(x) = u(a, b, s) \mid_{arguments}$, w.r.t s where $s = x_0$.

For completeness we write out the computer's steps to evaluate 3 under the conditions of 2c with a particular value of $s = x_0$, in equation format:

$$\frac{\partial}{\partial x} f(x) \mid_{x_0} = \frac{\partial u}{\partial v} \mid_{1,2,v|_{1,1,w|_{1,1,x_0}}} \cdot \frac{\partial v}{\partial w} \mid_{1,1,w|_{1,1,x_0}} \cdot \frac{\partial w}{\partial s} \mid_{1,1,x_0} \cdot \frac{ds}{dx}$$

Repeating above in tabular format:

Current Evaluation	s Value	Partial Derivative
$w(a, b, s) \mid_{1,1,s}$	x_0	$\frac{\partial w}{\partial s} \mid_{1,1,x_0}$
$v(a, b, s) \mid_{1,1,s}$	$w(a, b, s) \mid_{1,1,x_0}$	$\frac{\partial v}{\partial w} \mid_{1,1,w\|_{1,1,x_0}} \cdot \frac{\partial w}{\partial s} \mid_{1,1,x_0}$
$u(a, b, s) \mid_{1,2,s}$	$v(a, b, s) \mid_{1,1,w(a,b,s)\|_{1,1,x_0}}$	$\frac{\partial u}{\partial v} \mid_{1,2,v\|_{1,1,w\|_{1,1,x_0}}} \cdot \frac{\partial v}{\partial w} \mid_{1,1,w\|_{1,1,x_0}} \cdot \frac{\partial w}{\partial s} \mid_{1,1,x_0} \cdot 1$

Because the computer knew the analytical forms of the partial derivatives of each of u, v, w beforehand, all it needed to do was:

1. to evaluate each of u, v, w according to 2c, in order from $w \to v \to u$,
2. to remember the values for x_0 and the output of each function evaluation besides u,
3. then to use x_0 and the output of the function evaluations as input for the corresponding partial derivative function evaluations, and
4. store the individual partials.

Lastly, to compute the partial derivative of the entire composition function, the computer multiplies the individual partials together in observance of the chain-rule.

Some things to note about AD are that no approximation of derivatives is being made because the analytical forms of the partials of the elementary math functions are defined alongside them. Accuracy of AD is then limited by the precision of the AD system's definition of the elementary math functions and their partials.

2.2. Tangent-stiffness Matrices

This subsection gives a working definition of a tangent-stiffness matrix and background on a Newton-Raphson method which uses a tangent-stiffness matrix to help solve nonlinear systems of equations, similarly to the the solution method used by the software package used in the study.

2.2.1. What is a Tangent-stiffness Matrix

It is important to identify what a tangent-stiffness matrix is and what it is used for to help show the motivation for the work discussed here. It is helpful to begin with the thought that a tangent-stiffness matrix is a type of slope for vector valued functions of vector variables at a single point in the space of the vector valued variable. A tangent-stiffness matrix is similar to the "m" in the scalar function $y = mx + b$, except that y and x are vectors while m is a matrix.

In detail, a tangent-stiffness matrix (or operator) can be described as a collection of the first order partial derivatives of a vector valued function w.r.t each degree of freedom of the vector valued function for a given value of the vector variable. The tangent-stiffness matrix comprises a linear operator that can be used to transform a difference in the vector variable into a difference in the value of the vector valued function via an inner product between the tangent-stiffness matrix and the vector variable. As an example in a solid mechanics system, suppose some dependent variables represented force components of force vectors at nodes of discretization, and independent variables represented displacement components for those same nodes. Suppose that as is possible in peridynamic nodes each dependent variable is a function of all of the independent variables. For this situation, the tangent-stiffness matrix would be the first derivatives of every force component variable with respect to every displacement component variable, with the rows ordering the force components and the columns ordering the displacement components.

How is a tangent-stiffness matrix computed? The mathematical formula for a tangent stiffness matrix can be expressed in indicial notation as:

$$\frac{\partial F_i}{\partial X_j} \Big|_{X_0,...,X_n}$$

F_i is the i'th component of the vector valued function, X_j is the j'th component of the vector variable, and a particular value of the vector variable is chosen $X_0, ..., X_n$. One then evaluates the expression for each combination i, j corresponding to an element at $row, column$ in the tangent-stiffness matrix. By inspection, the elements of a tangent-stiffness matrix can be estimated using the CTSE, AD and finite-difference techniques for functions $F : R^1 \to R^1$, since taking partial derivatives entails holding all but a single independent variable of the function constant, and each component of a vector valued function can be evaluated independently of the other components.

For reference, expressions for calculating tangent stiffness matrices with CTSE, AD, central-difference and forward-difference will be shown and explained.

2.2.2. Solve a Nonlinear System with a Tangent-stiffness Matrix

Michael: ~~An accurate tangent-stiffness matrix is a main part of the Newton-Raphson method for iteratively solving non-linear systems for equilibrium solutions. The method is derived by a linearisation using Taylor expansion of the vector valued function modelling the system in a way reminiscent of how forward-difference is derived from the Taylor expansion of a scalar valued function. Here the two derivations are shown side by side: The difference between the two is that in the forward-difference derivation, it is assumed that the derivative term is unknown and can be determined from knowing change in location, while in the Newton-Raphson derivation, it is assumed that the change in location is unknown and can be determined from knowing the value of the derivative term. Additionally we could derive a version of Newton-Raphson for CTSE directly: The Newton-Raphson method is applied in the following manner: It is clear to see how if the tangent-stiffness matrix used in the Newton-Raphson method is inaccurate, predicted changes in location will be inaccurate. However, non-linear functions don't have constant slopes by definition, meaning that a perfectly accurate tangent-stiffness matrix at the current guess location is not a guarantee of a one step solution path to the solution location since slope changes over the interval guess to updated guess.~~ The reader is referred to a succinct explanation of the Newton Raphson method, which shows the role of a tangent-stiffness matrix or Jacobian in that solution method [7, chap. 13].

3. Methods and Materials

3.1. Material: Perdynamics Code

3.1.1. Perdigm *and Peridynamics*

3.1.2. Implementing Complex-Step in Peridigm

3.2. The Comparative Study

3.2.1. Justification: Goals, Assumptions, and Metrics

A goal of the study was to rank complex-step against forward-difference and central difference on the basis of accuracy. AD was ommited from the comparison because it served as the standard of accuracy for the other methods in the absence of appropriate analaytical forms for the Jacobian associated with the nonlinear system used solved in the study. The assumption that AD is accurate enough to serve as a standard is supported by AD's implementation as a computerized chain rule as explained in 2.1.3. For a concrete comparison of each of the methods including AD to a known analytical Jacobian, refer to the corresponding author's website, specifically the C++ source-code example *BeamAndSpringSystem*.

Because it makes no sense to compare tangent-stiffness matrices from different load steps, or from different iterations, it was necessary to solve one load step and conclude each iteration within that load step by updating guesses only from the the results of the AD method, and to subsequently feed all four methods the same updated guess the following iteration. This decision precluded a comparison of convergence rate, since if the methods were allowed to solve a problem at their invidual pace, differences in accuracy would produce differences in guess updates and therefor the number and nature of iterations perfomed. Different guesses from iterations started with different previous guesses could not be data for a valid comparison of tangent-stiffness accuracy between methods. However a separate convergence rate comparison study could be done given some modifications to the materials for the study described in this paper, and of course, the resources to perform that study. Instead, in this study, speed of iteration was measured because it could be so done at the same time as Accuracy was being measured given a single tangent-stiffness calculation, which was attractive due to limited resources. Additionally, having each of the methods physically operate in the same process, single or parallel, allowed the comparison of tangent-stiffness matrices as loaded from random access memory rather than from the hard-disk, giving the benefit of vastly greater speed and the benefit of simplicty compared to some other solution speculatively involving dynamic file management on files which for one of the components of the study would be on the order of $1XE11$ bytes, for which "vastly" is not hyperbole.

The other goal was to rank the four methods, that is including AD, on the basis of speed. It was assumed that the order that each method was evaluated was unimportant, that evaluating each method sucessively within each solver iteration did not affect their performance individually and that speed of computation did not change with time. These assumptions allowed the test program to run the same problem with each of the methods at effectively the same time and generate an equal volume of data from each method. It was also assumed that the Jacobian matrix calculation time for each of the methods did not vary based on what values the independent variables held, as they do from iteration to iteration as the guessed equilibrium solution is updated. This assumption allowed calculation time measurements to be averaged over all iterations within a solution attempt. The purpose of averaging calculation time measurements was to informally address the extraneous variables of evaluation order and computer system load due to other user's prcocesses or processes not associated with the study.

The goals of collecting accuracy and speed measurements were achieved by developing and implementing metrics within the simulation program used in the study. The metric used to measure the accuracy of a tangent-stiffness matrix was the Frobenius norm of the element-wise difference between the tangent-stiffness matrix produced by the method being evaluated and the tangent-stiffness matrix produced by the AD based method, given that both methods were set upon the same problem. The lower the value of this metric, the closer the other method's Jacobian was to the AD Jacobian, and therefore the more accurate the method was. The expression for the accuracy metric: $D = \sqrt{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(J_{AD}(i,j) - J_M(i,j))^2}$ Where D is distance, J_{AD} is the Jacobian matrix produced by the AD based method

3.2.2. Materials: Computers, Other Software

3.2.3. Methods: Reduction of Data

4. Results and Discussion

4.1. Speed Data

4.2. Accuracy Data

4.3. Efficiency Data

5. Conclusions and Further Work

5.1. Thoughts on Results

5.2. Thoughts on Perdigm

5.3. Final Thoughts on Complex-Step

6. Unfinished table

Full Test Matrix						
Cores	Target	Rows	horizon (m)	Nonzero Rows	Calc. Time (s) CS, CD, FD, AD	Accuracy Diff. (MPa) CS, CD, FD
1	1E3	3000	.3	1994454	3.5, 2.7, 1.7, 1.6	1.92E-10, 1.21E-4, .137
1	2E3	5400	.24	4252230	6.2, 4.9, 3.2, 2.9	2.28E-10, 9.94E-4, .148
1	3E3	8526	.21	7792128	11.2, 8.9, 5.7, 5.2	2.38E-10, 1.59E-4, .145
1	4E3	12288	.19	15389352	26.7, 21.0, 13.4, 12.2	2.33E-10, 4.61E-4, .12
1	6E3	18468	.16	20027466	28.1, 22.2, 14.4, 13.1	2.76E-10, 1.05E-3, .145
1	8E3	24000	.15	31388724	47.6, 37.7, 24.2, 21.9	2.64E-10, 1.65E-3, .133
1	1.2E4	33396	.131	40060026	55.6, 44.1, 28.4, 25.9	3.03E-10, 1.92E-3, .148
1	1.6E4	50700	.12	83778480	138.9, 109.6, 70.2, 64.0	3.63E-10, 1.64E-3, .123
1	3.2E4	96768	.095	165140640	277.3, 218.1, 139.4, 126.5	3.26E-10, 2.18E-3, .128
32	1E6	3E6	.03	1666968710	336.1, 277.7, 200.6, 233.1	6.21E-10, 1.52E-2, .176
64	1E6	3E6	.03	1666962752	169.9, 140.7, 102.0, 119.7	6.20E-10, 1.50E-2, .177
96	1E6	3E6	.03	1666760396	114.7, 95.0, 69.1, 79.7	6.18E-10, 1.50E-2, .177
128	1E6	3E6	.03	1666892660	86.4, 71.8, 52.4, 58.8	6.16E-10, 1.47E-2, .177

Appendix A. Extra

[1] S. Chapra, R. Canale, Numerical Methods for Engineers, sixth Edition, McGraw-Hill, inc., NY, 2010.

[2] A. Voorhees, H. Millwater, R. Bagley, Complex variable methods for shape sensitivity of finite element models, Finite elements in analysis and design 47 (10) (2011) 1146–1156. doi:10.1016/j.finel.2011.05.003.

[3] E. Phipps, D. Gay, R. Bartlett, Sacado: Automatic differentiation tools for c++ codes, Tech. Rep. SAND2009-7540C, Sandia National Laboratories: Computer Science Research Institute, presented at Trilinos User's Group Meeting 2009 in Albuquerue, New Mexico. (Nov 2009).

[4] M. Heroux, J. Willenbring, Trilinos users guide, Tech. Rep. SAND2003-2952, Sandia National Laboratories (2003).

[5] D. Gay, Semiautomatic differentiation for efficient gradient computations, Tech. Rep. SAND2004-4688P, Sandia National Laboratories: Optimization and Uncertainty Estimation Department (2004).

[6] A. Griewank, On automatic differentiation, Tech. Rep. CRPC-TR89003, Rice University: Center for Research on Parallel Computation (1989).

[7] T. Young, M. Mohlenkamp, Introduction to Numerical Methods and Matlab Programming for Engineers, Ohio University Department of Mathematics, OH, 2009.