A COMPARISON OF DIFFERENT METHODS FOR CALCULATING TANGENT-STIFFNESS MATRICES IN A MASSIVELY PARALLEL COMPUTATIONAL PERIDYNAMICS CODE

Michael Brothers

John T. Foster, Advisor

University of Texas at San Antonio

Acknowledgements

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

NETL
NATIONAL ENERGY TECHNOLOGY LABORATORY

TACC TEXAS ADVANCED COMPUTING CENTER
Powering Discoveries That Change The World

SIVIRT

Outline

-Research Objectives

-Derivation of CTSE

-CTSE and finite-difference, subtractive cancellation

-How to produce a tangent-stiffness (Jacobian) matrix

-Study on verification problem in illustrative code

-Scalability study in *Peridigm* [2] M.L. Parks, D.J. Littlewood, J.A. Mitchell, and S.A. Silling

Research Objectives

-To experimentally test the veracity of complex Taylor Series expansion (CTSE), also called complex-step, for computing Jacobian matrices

-Exam CTSE as an alternative to automatic-differentiation (AD) and finite-difference probing methods for tangent-stiffness (TS) calculation

-To profile the performance in context of Jacobian calculation methods specifically in a production scale, parallelized, code

Derivation of CTSE

Taylor expand in the complex plane

$$f(\beta_n + ih) = f(\beta_n) + \frac{\partial f(\beta_n)}{\partial \beta_n}\frac{ih}{1!} + \frac{\partial f^2(\beta_n)}{\partial \beta_n^2}\frac{(ih)^2}{2!} + \frac{\partial f^3(\beta_n)}{\partial \beta_n^3}\frac{(ih)^3}{3!} + \cdots$$

Taking the imaginary part of both sides and solving for the first derivative…

$$\frac{\partial f(\beta_n)}{\partial \beta_n} \approx \frac{\mathrm{Im}\left(f(\beta_n + ih)\right)}{h}$$

Derivation example adapted from work by John Foster and Eric Breseno based upon: Squire and Trapp [3]

Analytical derivative example

Find the derivative of the cosine function…

$$f(x) = \cos(x)$$

Perturb along the imaginary axis

$$f(x+ih) = \cos(x+ih)$$

Trig expand

$$f(x+ih) = \cos(x)\cosh(h) - i\sin(x)\sinh(h)$$

Take imaginary part, divide by h...

Analytical derivative example (contd.)

Apply the previously defined approximation.

$$\frac{\partial f(x)}{\partial x} = -\frac{\sin(x)\sinh(h)}{h}$$

Let $h \rightarrow 0$

$$\frac{\partial f(x)}{\partial x} = \lim_{h\to 0} -\frac{\sin(x)\sinh(h)}{h}$$

Apply L'Hôpital's Rule

$$\frac{\partial f(x)}{\partial x} = \lim_{h\to 0} -\frac{\sin(x)\cosh(h)}{1}$$

Finally...

$$\frac{\partial f(x)}{\partial x} = -\sin(x)$$

CTSE and FD, subtractive cancellation

Any time you subtract two [floating point] *numbers* [in a computer program] *which are almost equal, subtractive cancellation will occur, and the difference will not have as much precision as either of the subtrahend or the minuend. [4] –*

Douglas Wilhelm Harder, Department of Electrical and Computer Engineering,University of Waterloo

Subtractive cancellation [4]:

$$4 \text{ digit mantissa; } x_0 = 3.253, F'(x_0) = \mathbf{6.505}$$

$$\text{Forward Difference: } F'(x_0) \approx \frac{F(x_0+h) - F(x_0)}{h}$$

$$F(x) = x^{2.0}$$

$$\frac{3.255^{2.0} - 3.253^{2.0}}{.002} =$$

$$\frac{10.60 - 10.58}{.002} = \mathbf{10}$$

$$10 \neq 6.505$$

32-bit floating point, 7 digit mantissa; $x_0 = 1.5, F'(x_0) = 3.62203$

$$1 : CD : F'(x_0) \approx \frac{F(x_0+h)-F(x_0-h)}{2h}$$

$$2 : CTSE : F'(x_0) \approx \frac{Im(F(x_0+ih))}{h}$$

$$3 : F(x) = \frac{e^x}{\sin(x)^3+\cos(x)^3}$$

$$stepsize : h$$

h	Equation 1	Equation 2
.1000000E-01	0.362298E+01	0.362109E+01
.1000000E-02	0.362229E+01	0.362202E+01
.1000000E-03	0.362158E+01	0.362203E+01
.1000000E-04	0.360012E+01	0.362203E+01
.1000000E-05	0.357628E+01	0.362203E+01
.1000000E-06	0.476837E+01	0.362203E+01
.1000000E-07	0.000000E+00	0.362203E+01
.1000000E-08	0.000000E+00	0.362203E+01
.1000000E-09	0.000000E+00	0.362203E+01
.1000000E-10	0.000000E+00	0.362203E+01

Example from: Squire and Trapp [3]

CTSE resists subtractive cancellation

The how is simple:

-no difference is taken as in finite-differencing

$$1 : CD : F'(x_0) \approx \frac{F(x_0+h)-F(x_0-h)}{2h}$$

$$2 : CTSE : F'(x_0) \approx \frac{Im(F(x_0+ih))}{h}$$

How to produce a tangent-stiffness (Jacobian) matrix

$$\text{Analytical: } K_{ij} = \frac{\delta F_i^{int}(x)}{\delta x_j}$$

$$\text{FD: } K_{ij} = \frac{F_i^{int}(x+he_j) - F_i^{int}(x)}{h}$$

$$\text{CD: } K_{ij} = \frac{F_i^{int}(x+he_j) - F_i^{int}(x-he_j)}{2h}$$

$$\text{CTSE: } K_{ij} = Im\left(\frac{F_i^{int}(x+he_j^{imaginary})}{h}\right)$$

* 'int' signifies derivative of internal force WRT displacement

How the Jacobian is used to solve a force equilibrium problem: algorithm of illustrative code

Inputs: Function pointer for model evaluation, solver parameters

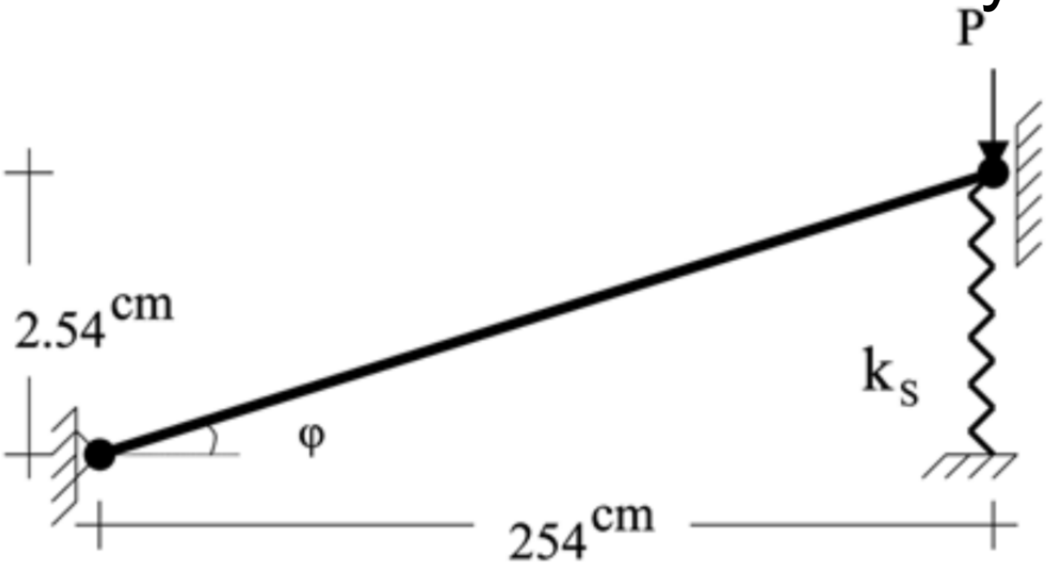
1)Setup:
   a) Define residual
   b) Define convergence criteria

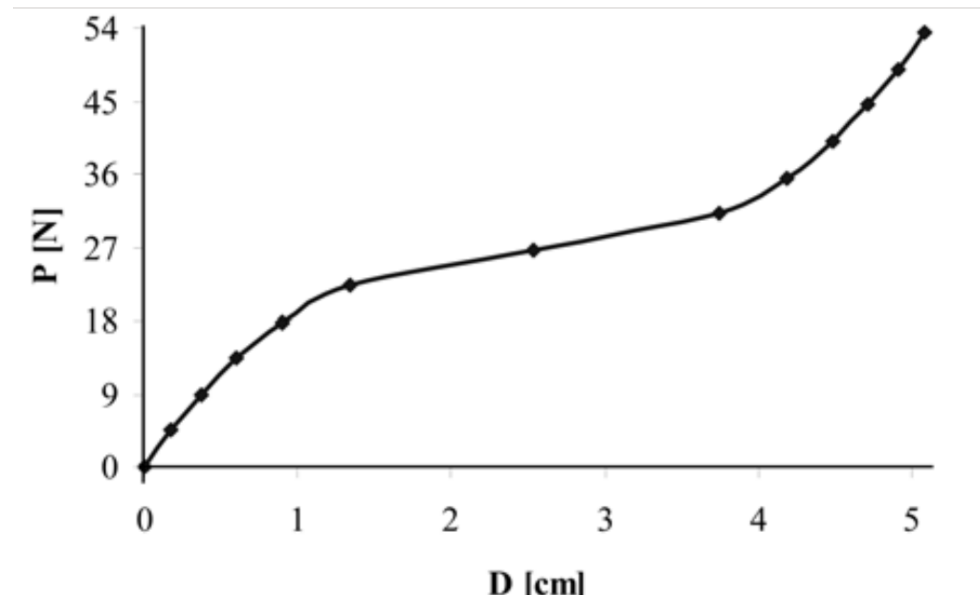
2) Main loop:
   a) Guess equilibrium solution
   b) Compute Jacobian
   c) Compute Newton update using LAPACK GETRF for LU factorization and LAPACK GETRS to solve.
   d) Compute change in residual
   e) Evaluate convergence criteria

3) Report last configuration and performance data

4) Finish or choose another target as for a subsequent load-step

Case study: verification





-Illustrative code implemented with data structures and wrappers from *Trilinos* library [5]

-Serves double duty as simple example implementation of each method, relative to *Peridigm* code

-Verification problem taken from Rezaiee Pajand, M., Alamatian, J. [6] and results reproduced for each method

Case study: verification: problem setup

$$f(D) = 0.5AE(\mathrm{Cos}^2\varphi)\left(\frac{D}{L_0}\right)^2\left[\frac{D}{L_0}\mathrm{Cos}^2\varphi - 3\mathrm{Sin}\,\varphi\right] + k_s D + \left(AE\frac{D}{L_0}\right)\mathrm{Sin}^2\varphi$$

$$S_T = 1.5AE(\mathrm{Cos}^2\varphi)\left[\frac{D}{L_0}\mathrm{Cos}^2\varphi - 2\mathrm{Sin}\,\varphi\right]\left(\frac{D}{L_0^2}\right) + k_s + \frac{AE\mathrm{Sin}^2\varphi}{L_0}$$

-Force function and analytical Jacobian given

-Apply force boundary condition in 12 load steps, each an Increment of 4.448 N, vary step-size h to measure accuracy response

-Verification is suggested when using a method results in minimum discrepancy wrt 'analytical' solution

Case study: verification: results

Displacement, D, after load-step 12				
Step exponent	Actual cm	AD cm	CS cm	FD cm
0	5.07996	"	"	"
-4	5.07996	"	"	"
-8	5.07996	"	"	"
-12	5.07996	"	"	"
-16	5.07996	"	"	failed
-20	5.07996	"	"	failed

Case study: convergence rate

-Same problem and code as before, now measuring number of iterations taken, not quality of solution

-Step-size is varied to measure effect on convergence rate

-Ideal convergence rate is supposed as the number of iterations the 'analytical' solution method takes.

-Study meant to give context to *Peridigm* iteration speed results, informally

Case study: convergence rate: results

Displacement, D, after load-step 12				
Step exponent	Actual cm	AD cm	CS cm	FD cm
0	63	"	213	417
-4	63	"	"	"
-8	63	"	"	"
-12	63	"	"	65
-16	63	"	"	fail
-20	63	"	"	fail

SIVIRT

Implementing CTSE in: *Peridigm*

-Change data-types where possible, cast everywhere else

-Modify copied finite difference model evaluation code to follow CTSE formula

-Add logic to select CTSE for use

-Instrument code for performance comparison

-Compare CTSE, CD, FD, AD for a series of test problems each for single-core and multi-core cases

Scope Stats:

-Refactored code:
3137 lines

-Workflow scripts:
2097 lines

-1 sample CPU load:
2280 hours
(¼ yr on 1 core)

-Simulations:
36 completed

Representative modification: calculating the displacement of a node

-All finite difference methods

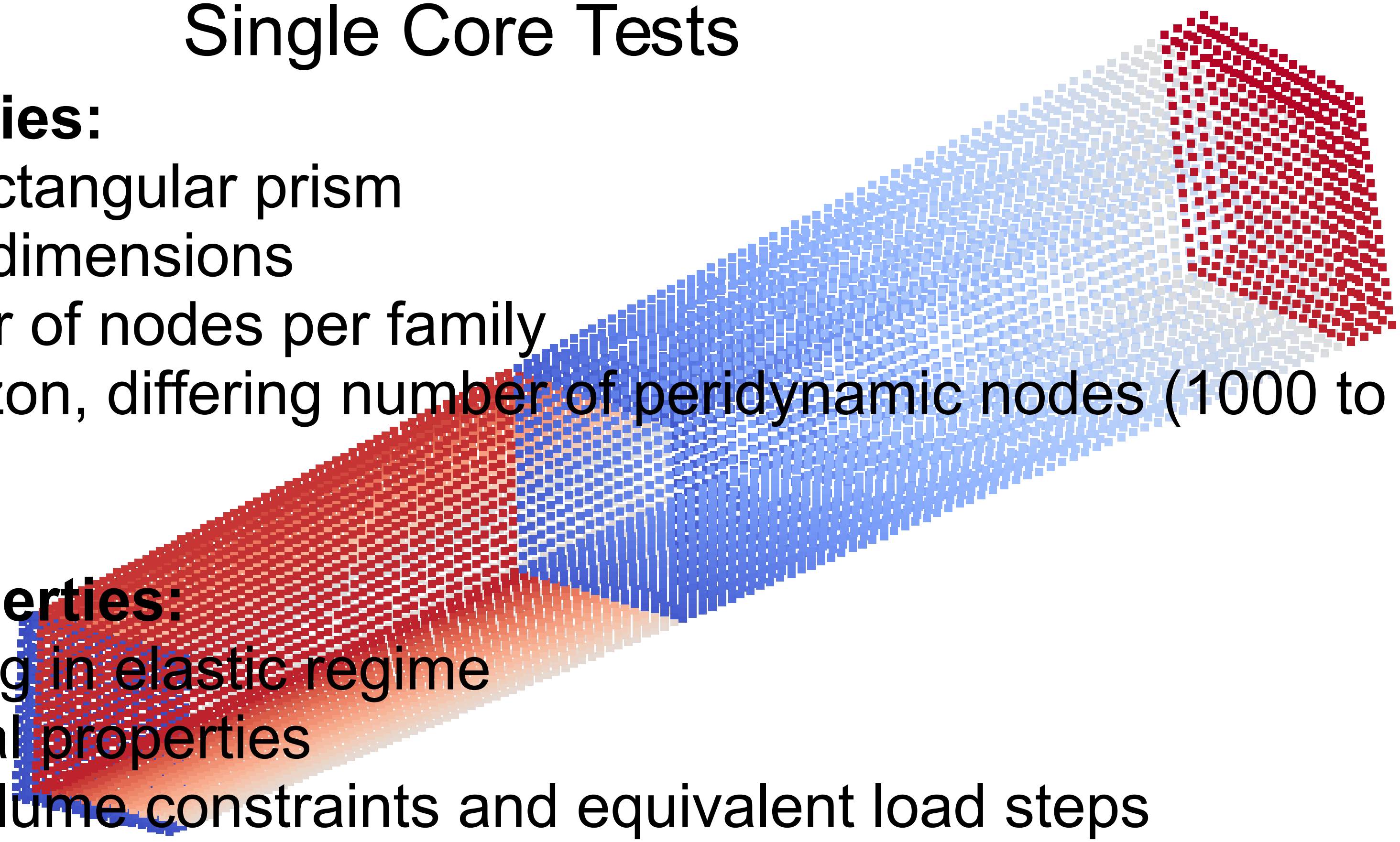
```
X_dx = XP[0]-X[0];
X_dy = XP[1]-X[1];
X_dz = XP[2]-X[2];
zeta = sqrt(X_dx*X_dx+X_dy*X_dy+X_dz*X_dz);
Y_dx = YP[0]-Y[0];
Y_dy = YP[1]-Y[1];
Y_dz = YP[2]-Y[2];
dY = sqrt(Y_dx*Y_dx+Y_dy*Y_dy+Y_dz*Y_dz);
```

-CTSE: trivially more complicated

```
X_dx = XP[0]-X[0];
X_dy = XP[1]-X[1];
X_dz = XP[2]-X[2];
zeta = sqrt(X_dx*X_dx+X_dy*X_dy+X_dz*X_dz);

Y_dx = std::complex<double>(YPREAL[0]-YREAL[0], YPIMAGINARY[0]-YIMAGINARY[0]);
Y_dy = std::complex<double>(YPREAL[1]-YREAL[1], YPIMAGINARY[1]-YIMAGINARY[1]);
Y_dz = std::complex<double>(YPREAL[2]-YREAL[2], YPIMAGINARY[2]-YIMAGINARY[2]);
dY = sqrt(Y_dx*Y_dx+Y_dy*Y_dy+Y_dz*Y_dz);
```

Single Core Tests



Mesh Properties:

-Modeled a rectangular prism
-Same object dimensions
-Same number of nodes per family
-Differing horizon, differing number of peridynamic nodes (1000 to 32000 nodes)

Loading Properties:

-Tensile loading in elastic regime
-Same material properties
-Equivalent volume constraints and equivalent load steps

Notes on Operation:

-Each Jacobian calculation method uses same displacement guess
-Speed, accuracy and efficiency are compared between AD, FD, CD and CTSE

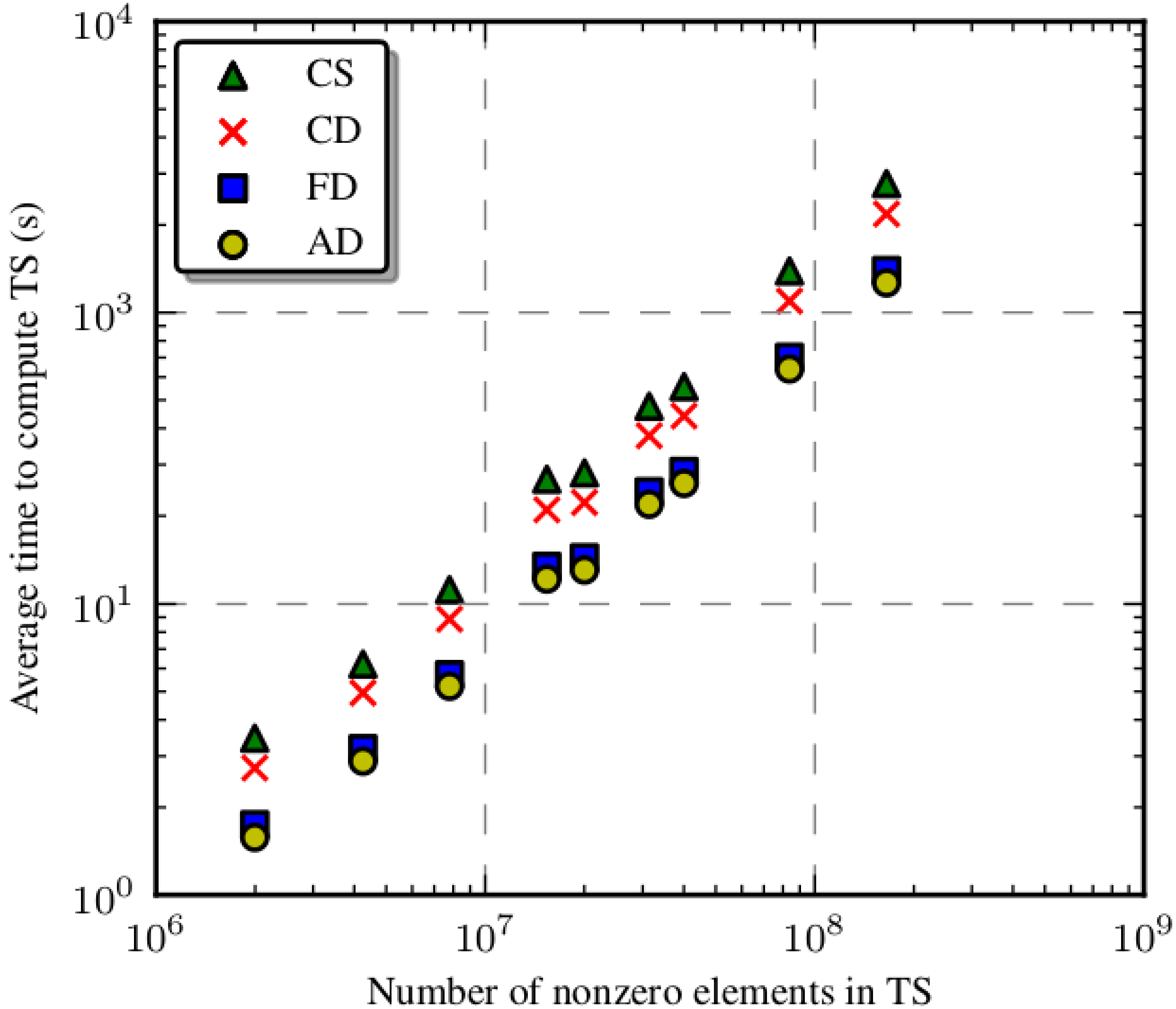


Figure 3.1: Serial test series speed measurements.

SIVIRT

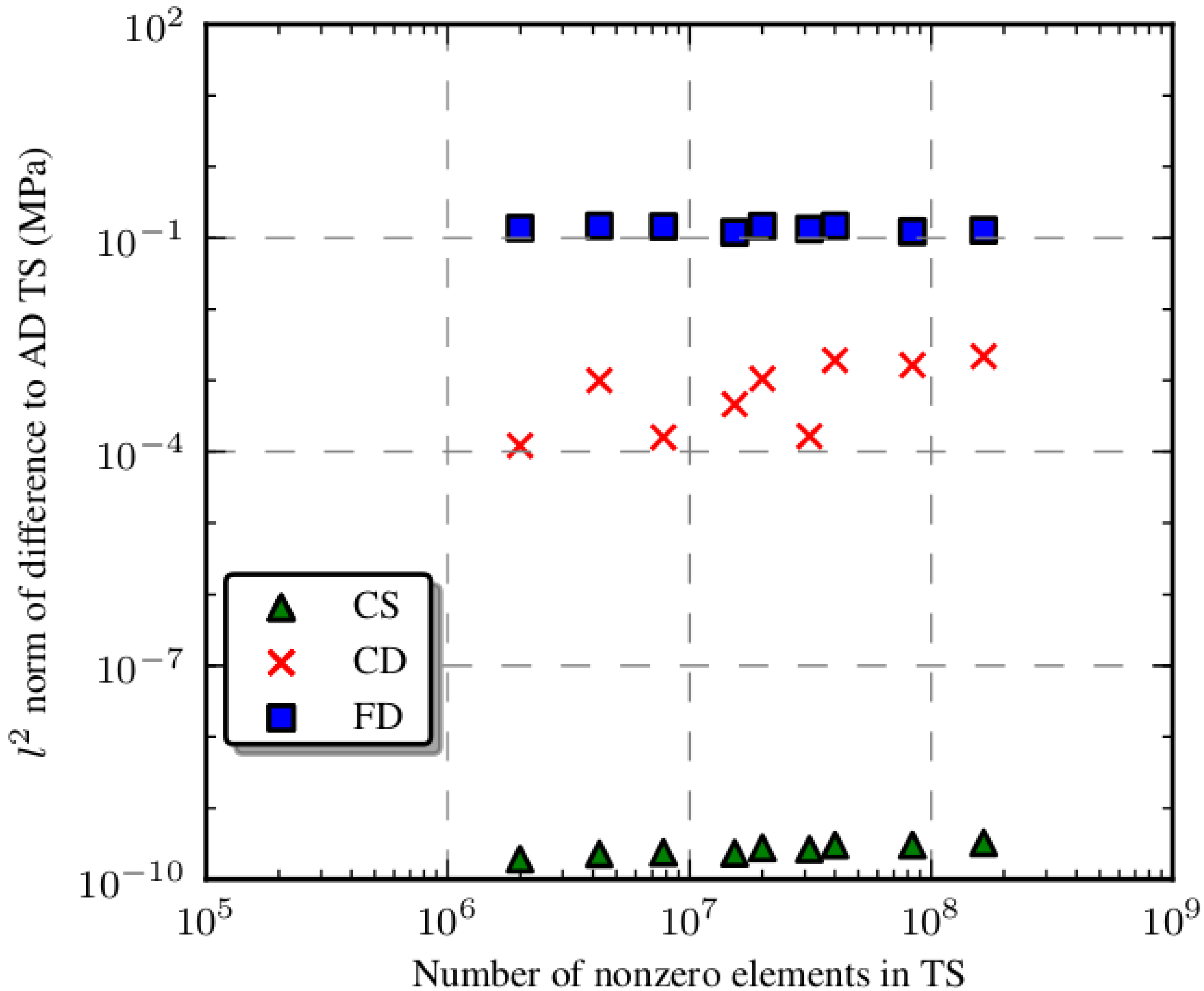


Figure 3.3: Serial test series accuracy measurements.

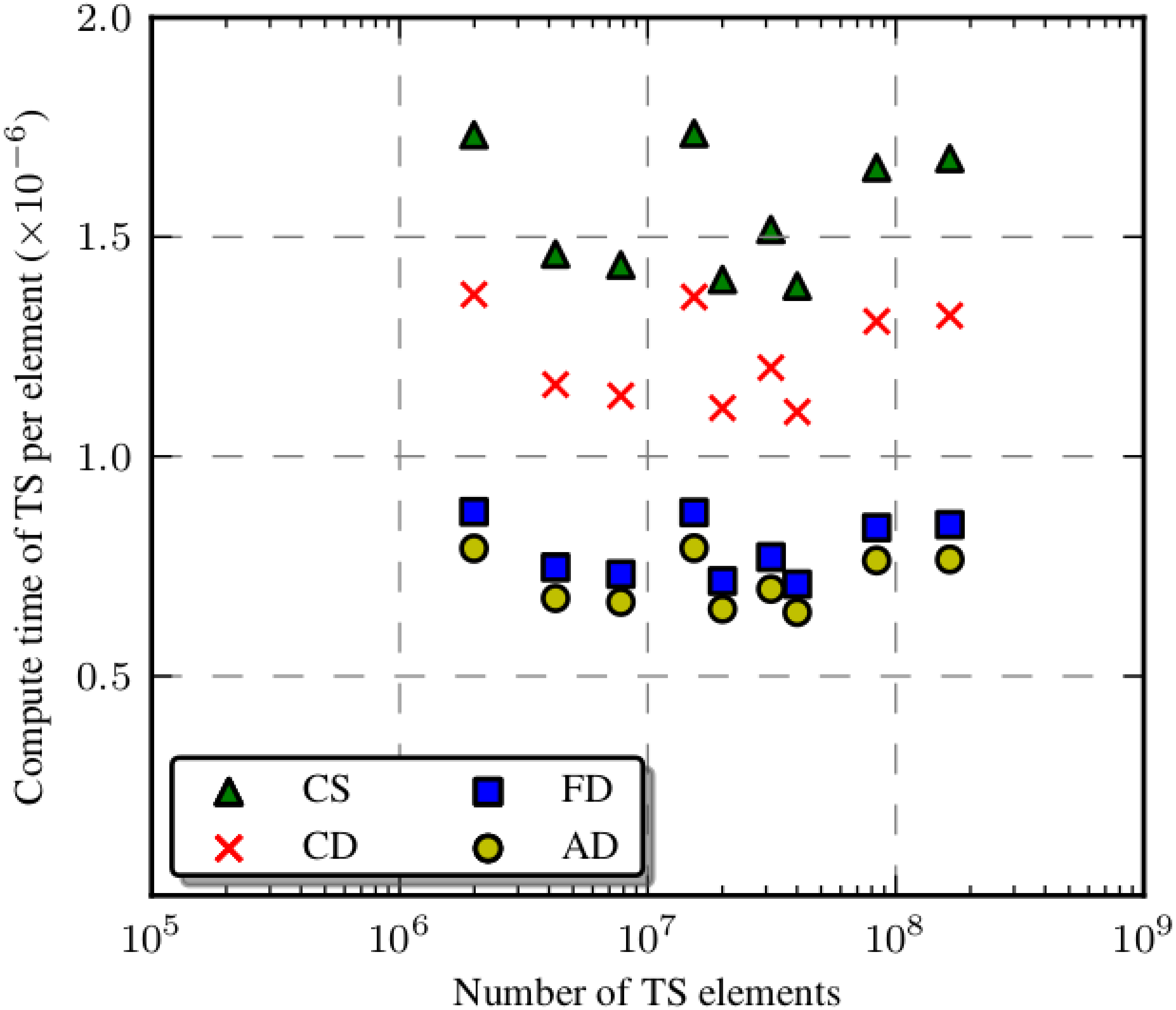


Figure 3.5: Serial test series efficiency measurements.

Multi-core Tests

Mesh Properties:

-Modeled a rectangular prism

-Same object dimensions

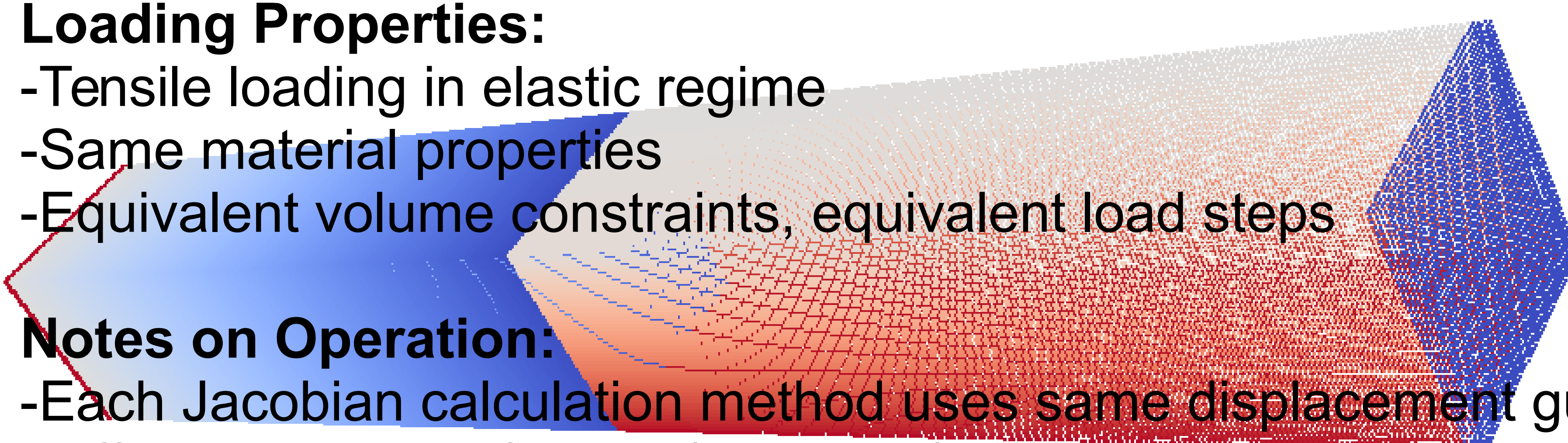
-Same horizon, same number of peridynamic nodes (3 million dof)

Loading Properties:

-Tensile loading in elastic regime

-Same material properties

-Equivalent volume constraints, equivalent load steps



Notes on Operation:

-Each Jacobian calculation method uses same displacement guess

-Differing number of cores (32 to 128) used to solve problem, with mesh decomposition differing

-Speed, accuracy and efficiency are compared between AD, FD, CD and CTSE

SIVIRT

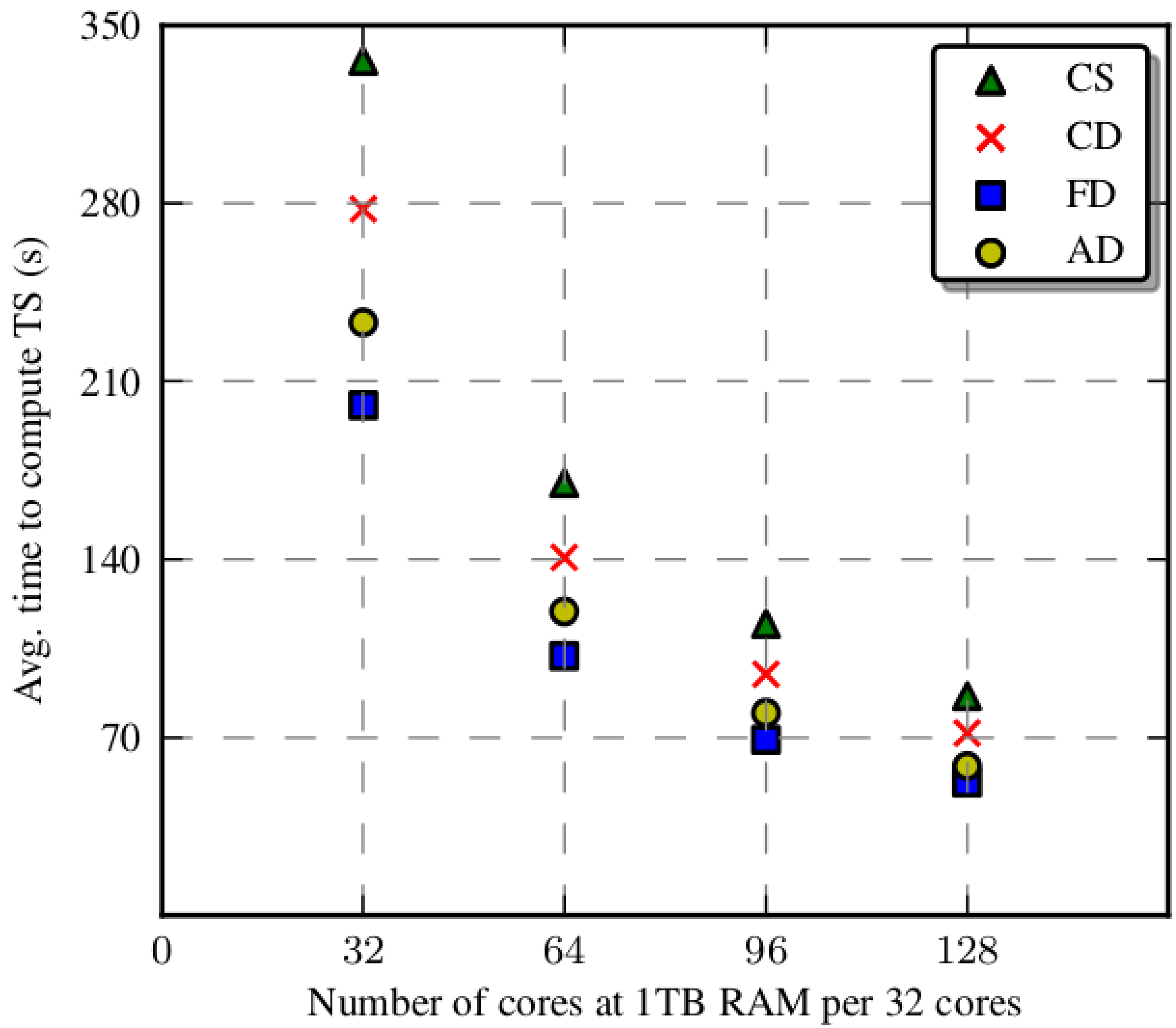


Figure 3.2: Multicore test series speed measurements.

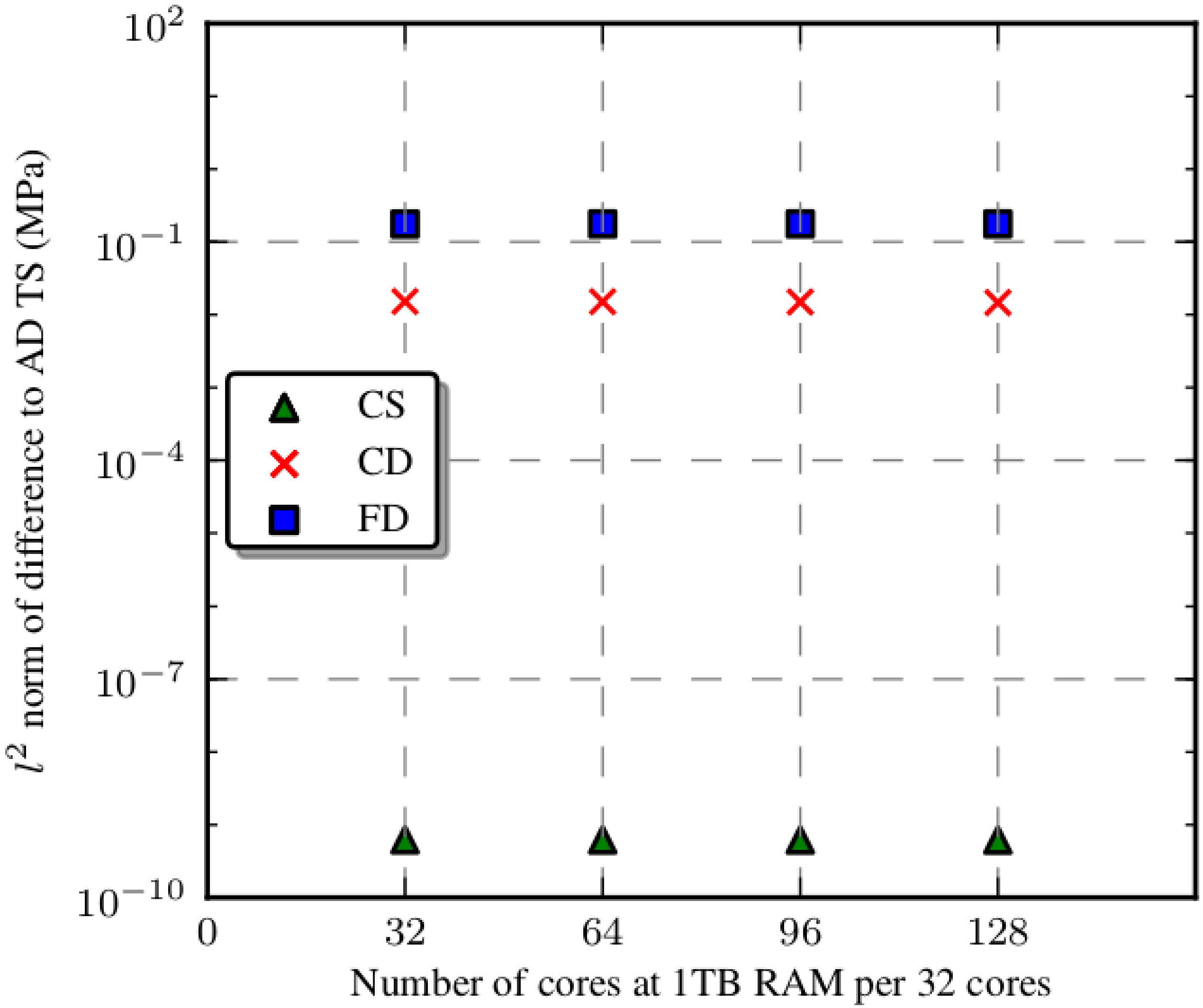


Figure 3.4: Multicore test series accuracy measurements.

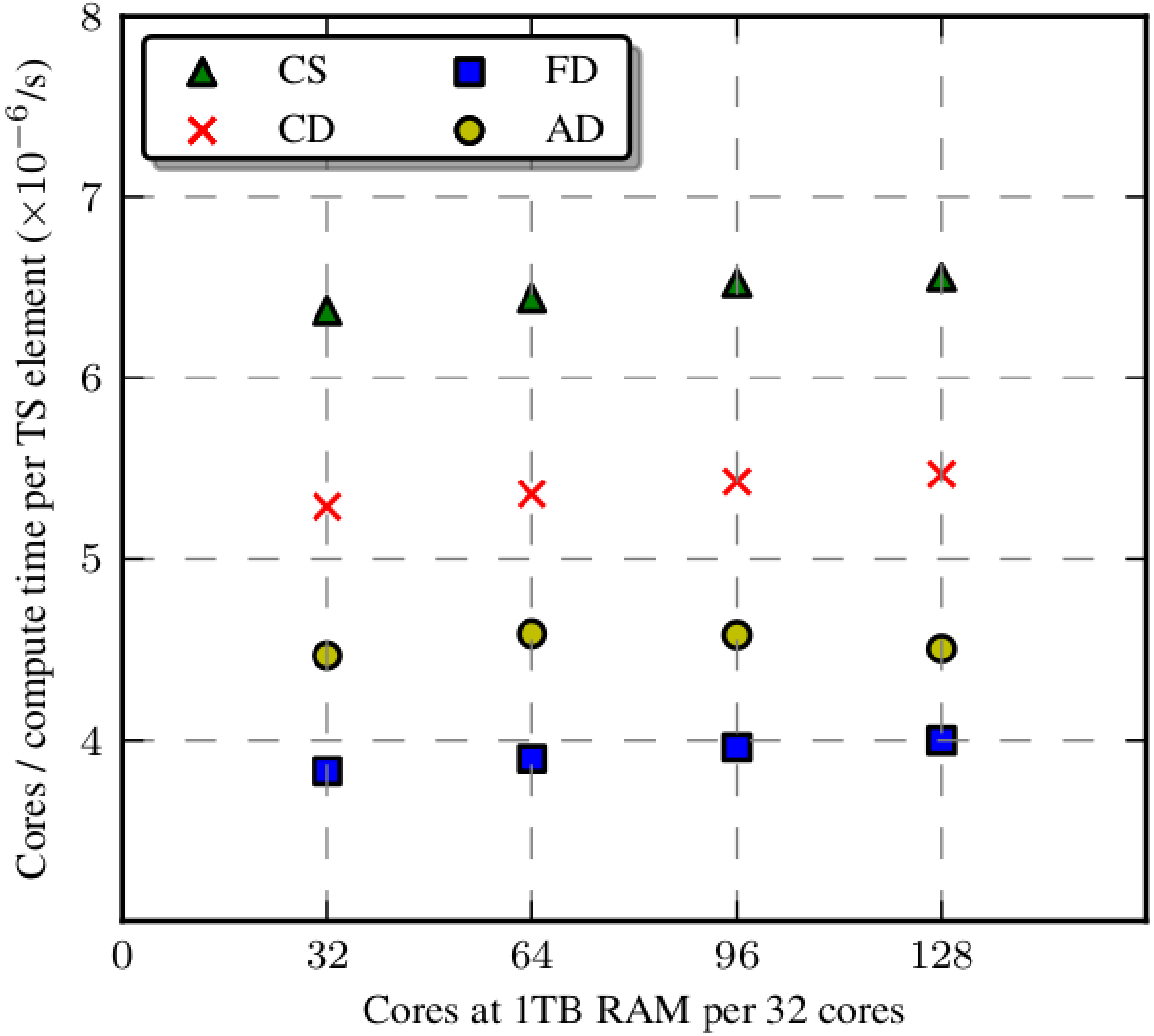


Figure 3.6: Multicore test series efficiency measurements.

Conclusions

Numeric benefits of CTSE:
-high accuracy Jacobians
-resists subtractive cancellation
-fast convergence rate

Practical, but abstract benefits of CTSE:
-requires fewer external libraries than AD does
-native, byte-copyable data-types
-easier for programmer to understand than AD

Drawbacks of CTSE:
-requires complex data-type
-slow iteration speed in a normal implementation

References

[1] Voorhees, Andrew, Harry Millwater, and Ronald Bagley. "Complex Variable Methods for Shape Sensitivity of Finite Element Models." Finite Elements in Analysis and Design 47 (2011): 1146-156. Print.

[2] M.L. Parks, D.J. Littlewood, J.A. Mitchell, and S.A. Silling, Peridigm Users' Guide, Tech. Report SAND2012-7800, Sandia National Laboratories, 2012.

[3] Squire, William, and George Trapp. "Using Complex Variables to Estimate Derivatives of Real Functions." SIAM Review 40.1 (1998): 110. Print.

[4] Harder, Douglas W. "Numerical Analysis for Engineering." Weaknesses with Floating-point Numbers. Department of Electrical and Computer Engineering: University of Waterloo, n.d. Web. 18 June 2013.

References

[5] Heroux, M., Bartlett, R., Howle, V., Hoekstra, R., Hu, J., Kolda, T., Lehoucq, R., Long, K., Pawlowski, R., Phipps, E., Salinger, A., Thornquist, H., Tuminaro, R., Willenbring, J., Williams, A., Stanley, K.: An overview of the trilinos project. ACM Trans. Math. Softw. 31(3), 397–423 (2005). DOI http://doi.acm.org/10.1145/1089014.108902

[6] Rezaiee Pajand, M., ALAMATIAN, J.: The dynamic relaxation method using new formulation for fictitious mass and damping. Structural Engineering and Mechanics 34 (2010)

Data and experimental materials are available at:
http://idl.utsa.edu/jtfoster/paper-repositories/