Adagio: non-linear quasi-static structural response using the SIERRA framework

John A. Mitchell *, Arne S. Gullerud, William M. Scherzinger, Richard Koteras, Vicki L. Porter

Sandia National Laboratory, P.O. Box 5800, MS 0847, Albuquerque, NM 87185, USA

Abstract

Adagio is a quasistatic nonlinear finite element program for use in analyzing the deformation of solids. It is massively parallel, built upon the SIERRA finite element framework [1], and employs the ACME library [2] for contact search algorithms.The mechanics and algorithms in Adagio closely follow those previously developed in JAC2D by Biffle and Blanford [3] as well as JAS3D by Blanford et al. [4]. Adagio assumes a quasistatic theory in which material point velocities are retained but time rates of velocities are neglected. Sources of nonlinearities include nonlinear stress–strain relations, large displacements, large rotations, large strains, and frictional/frictionless contact mechanics. Quasistatic equilibrium is found using a nonlinear solution strategy which includes nonlinear conjugate gradients. This paper briefly describes quasistatic equilibrium, kinematics of deformation, stress updates and the nonlinear solution strategy used in Adagio. In addition, we briefly describe how Adagio is implemented within the SIERRA architecture. Finally, we demonstrate Adagio's massively parallel capabilities on an example problem.

Keywords: Nonlinear solid mechanics; Quasistatics; Conjugate gradients; Object oriented

1. Quasistatic equilibrium

Quasistatic equilibrium in Adagio is based upon the principle of virtual work in a rate form. We start by writing down a nonlinear functional representing the power input to the body in the current configuration. By taking the first variation of the power input, and integrating by parts, we arrive at the weak form:

$$\int_{\Omega} \overline{T} : \delta\dot{\overline{\varepsilon}}\, \mathrm{d}V - \int_{\Omega} \rho\vec{b} \cdot \delta\vec{v}\, \mathrm{d}V - \int_{\partial\Omega} \vec{t} \cdot \delta\vec{v}\, \mathrm{d}A = 0 \qquad (1)$$

where Ω corresponds to the volume of the body in the current configuration, $\partial\Omega$ is the boundary of the body in the current configuration,$\overline{T}$ is the Cauchy stress tensor, $\vec{v}$ is the material point velocity, $\delta\dot{\overline{\varepsilon}}$ is the symmetric part of the virtual velocity gradient, $\vec{t}$ is an applied surface traction, ρ is mass density, and $\vec{b}$ is a body force vector. The extent to which Eq. (1) is not satisfied is a measure of the force imbalance and lack of quasistatic equilibrium. This force imbalance is called the residual and quasistatic equilibrium is defined according to how close the residual is to zero.

Adagio solves for quasistatic equilibrium over a set of time increments $\Delta t = t_{n+1} - t_n$ defined by a sequence of times $t_n \quad n = 0, 1, 2, \ldots$. A force imbalance occurs at t_{n+1} due to loads, temperatures, or kinematic boundary conditions that are parameterized by time. Quasistatic equilibrium is assumed to exist at t_n. The solver searches for a suitable equilibrium configuration at t_{n+1} through a sequence of trial velocities that give rise to ever decreasing residuals (force imbalance). Equilibrium is satisfied when the force imbalance reaches a user specified tolerance for convergence.

2. Updated Lagrangian

The solver finds velocity vectors $\vec{v}_{n+1}$ for a load step at discrete times t_{n+1} by solving the nonlinear problem

* Corresponding author. Tel.: +1 (505) 844-3435; Fax: +1 (505) 844-9297; E-mail: jamitch@sandia.gov

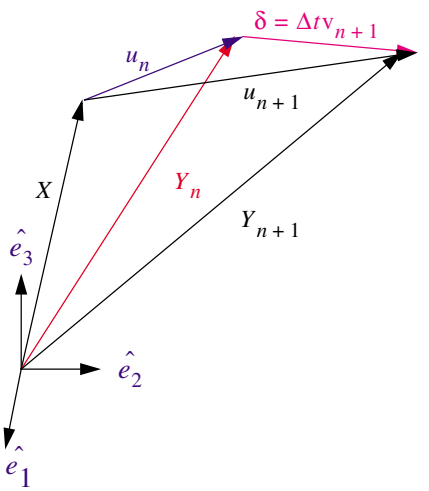

Fig. 1. Updated Lagrangian schematic.

implied by the weak form (1). The current position $\vec{y}_{n+1}$ of material points is updated via the formula:

$$\vec{y}_{n+1} = \vec{X} + \vec{u}_n + \Delta t \vec{v}_{n+1} \tag{2}$$

where $\vec{X}$ are the material coordinates at $t = 0$, $\vec{u}_n$ is the total displacement at the last converged step t_n, and $\Delta t = t_{n+1} - t_n$ is the time step size taken for the load step. This updated Lagrangian approach is depicted in Fig. 1.

3. Kinematics of deformation

In order to manage a variety of constitutive models as well as large rotations in conjunction with objective stress rates, we calculate a total deformation gradient $F = \partial \vec{y}_{n+1}/\partial \vec{X}$ and an incremental deformation gradient $\hat{F} = \partial \vec{y}_{n+1}/\partial \vec{y}_n$. However, we usually work with the inverse deformation gradients $F^{-1} = I - (\partial \vec{u}_{n+1}/\partial \vec{y}_{n+1})$ for purposes of computational efficiency since we need to evaluate the internal force vector that requires gradient and divergence operations in the current configuration (see Eq. (1)). Using the polar decomposition theorem on the deformation gradients, we calculate rates of strain, total stretches, and rotation operators. The polar decomposition on the deformation gradients is defined as $F = RU$ and for the inverse deformation gradients it is defined as $F^{-1} = R^t V^{-1}$, where R is an orthogonal rotation operator, and U and V are the corresponding stretch tensors. The incremental deformation gradients are similarly decomposed and are used for purposes of calculating rates of strain at material points for hypoelastic material models. Total deformation gradients are used for managing tensors in unrotated and rotated configurations as shown in Fig. 2 as well as in hyperelastic constitutive models which require a measure of the total strain.

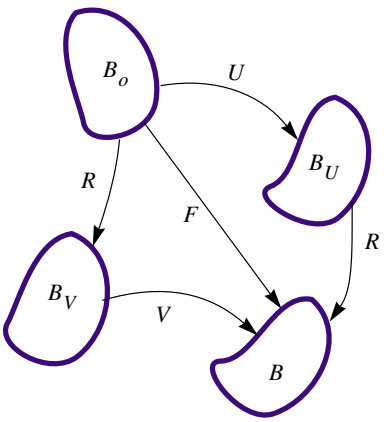

Fig. 2. Kinematics of the deformation.

4. Stress rate and hypoelastic stress updates

Most material models in Adagio are hypoelastic so that stress rates are integrated forward in time over the time step $\Delta t = t_{n+1} - t_n$ to find the stress at t_{n+1}. In order to develop this methodology, we first define an unrotated cauchy stress (configuration B_U) $\overline{\sigma} = R^t \overline{T} R$. The unrotated cauchy stress rate which is analogous to the Green–Naghdi stress rate is objective and defined abstractly by $\dot{\overline{\sigma}} = f(\Delta t, d, \overline{\sigma})$, where $f(d, \overline{\sigma})$ represents the incremental form of the constitutive model specifics. The Green–Naghdi stress rate is defined as:

$$\hat{\sigma} = R \dot{\overline{\sigma}} R^t = \dot{\overline{T}} - \Omega \overline{T} + \Omega \overline{T} \tag{3}$$

where $\Omega = \dot{R} R^t$, and T is the cauchy stress tensor. Our algorithm for updating stresses is given as: (1) compute strain rate $D = -\frac{1}{\Delta t} \ln \hat{V}^{-1}$; (2) de-rotate strain rate using $d = R^t D R$; (3) integrate constitutive model $\dot{\overline{\sigma}} = f(\Delta t, d, \overline{\sigma})$ to find $\overline{\sigma}$; (4) rotate $\overline{\sigma}$ to current configuration $\overline{\sigma} = R^t \overline{T} R$. Note that $\hat{V^{-1}}$ is the incremental left stress tensor.

5. Solution strategy: nonlinear PCG

The primary method for finding quasistatic equilibrium in Adagio is the preconditioned conjugate gradient method (PCG) [5]. The solver is configured in an object oriented way and consists of the following abstract plugins: preconditioner, line search, and residual operator. Just prior to running the solver for each loadstep, a loadstep predictor is invoked. The predictor runs a line search with the velocity vector from the last converged loadstep as the search direction.

5.1. Preconditioning

The process of solving a loadstep with PCG is an incremental solution strategy and is conceptually very similar to

Newton–Raphson. We construct a preconditioner B that is an approximation to ${K_t}^{-1}$ (inverse of tangent stiffness). In addition, the solver requires that the output of the preconditioner satisfy the homogeneous form of kinematic boundary conditions as well as contact constraints. Currently, Adagio has a nodal preconditioner which consists of a three-by-three block diagonal stiffness for each node in the mesh. These stiffnesses may be computed by probe or through an analytical formula.

5.2. Line search

Both the PCG solver and the predictor in Adagio use a line search object as a plugin. The PCG algorithm assumes that the line search produces a new velocity such that the resulting residual will be orthogonal to the current search direction. This corresponds to an exact line search. However, this is usually a multi-step process and can be expensive. Adagio currently performs an inexact line search by using one step of a secant line search algorithm. This line search is used in both the predictor and solver.

5.3. Residual operator

Quasistatic equilibrium is fundamentally based upon the residual/force imbalance. The PCG solver, predictor and line search objects in Adagio all use residual operators. In Adagio, the residual operator is responsible for managing geometry, external forces, internal forces and reaction forces on surfaces where kinematic boundary conditions are applied.

6. Code architecture

The algorithms in Adagio described above are implemented within the SIERRA framework. SIERRA-based codes consist of mechanics modules which can be nested inside each other to provide a rational code hierarchy. Fig. 3 depicts the overall architecture of Adagio. The highest level of control in Adagio is Agio_Procedure, which manages time stepping. Nested inside of Agio_Procedure is Agio_Region, which is responsible for orchestrating all calculations required for a particular time step. Agio_Region contains a set of mechanics modules that perform individual algorithms and are dynamically loaded at run time based upon user input. Fig. 3 shows several examples of these mechanics. They include: Agio_KinBC, which computes the effects of boundary conditions; Elements, which conducts element computations; Agio_NonlinearPCG, which drives the solver; and Agio_Predictor, which provides a predicted first guess for the solver. The nesting continues within the solver and the predictor. Any of the mechanics shown in Fig. 3 can be replaced by a different mechanics module as long as it conforms to the minimal interface as shown.

The runtime behavior of Adagio is closely tied to the construction, scoping, and registration of algorithms on mechanics modules. The concept of scope in SIERRA is somewhat analogous to that of C++. For example, in Fig. 3 Agio_Fe_Operator exists in two locations: Agio_LineSearch and Agio_NonlinearPCG. These two operators are at different scope and may have totally different implementations. Mechanics algorithms in SIERRA are invoked via a constant interface and algorithms run according to whether they are in the current scope. For example, dur-

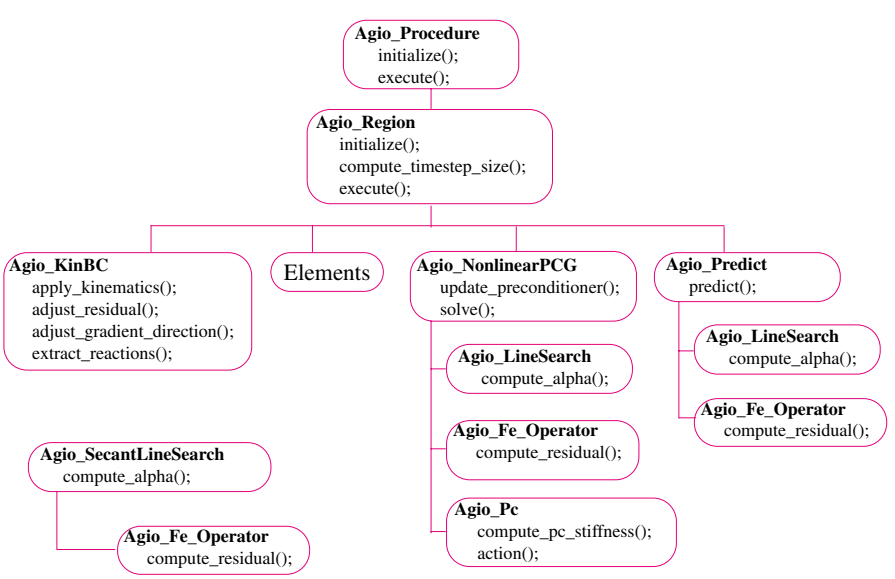

Fig. 3. Schematic of Adagio mechanics algorithms.

ing the solution process, Agio_NonlinearPCG invokes the algorithm "compute_residual" once per iteration. Any mechanics nested within Agio_NonlinearPCG which has the algorithm "compute_residual" will have its algorithm executed — in this case, only Agio_Fe_Operator has an algorithm that will be executed. The Agio_Fe_Operator within Agio_LineSearch will not get executed. Furthermore, if no Agio_Fe_Operator was registered inside Agio_NonlinearPCG, then nothing would happen when "compute_residual" is called. These mechanisms provide significant power for Adagio to connect modules with similar but not identical behavior, and to create a logical code structure where functionality can be selected easily.

References

[1] Edwards C, Stewart JR. SIERRA: A software environment for developing complex multi-physics applications. In: Bathe KJ, First MIT Conference on Computational Fluid and Solid Mechanics, 2001. Amsterdam: Elsevier Science.

[2] Brown KH, Glass MW, Gullerud AS, Heinstein MW, Jones RE, Summers RM. ACME: A parallel library of algorithms for contact in a multi-physics environment. In: Bathe KJ, First MIT Conference on Computational Fluid and Solid Mechanics, 2001. Amsterdam: Elsevier Science.

[3] Biffle JH, Blanford ML. JAC2D: A two-dimensional finite element computer program for the nonlinear quasi-static response of solids with the conjugate gradient method. Sandia National Laboratories, Albuquerque, NM. SAND93-1891, 1994.

[4] Blanford ML, Heinstein M, Key SW. JAS3D: A multi-strategy iterative code for solid mechanics analysis. Users' Instructions, Release 1.6. Sandia National Laboratories, Albuquerque, NM. To be published as a SAND report, 2000.

[5] Shewchuk JR. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1994. Edition 1.25.