Software Requirements for Tempus: Time Integration Package

Curtis C. Ober, Multiphysics Applications
Roger P. Pawlowski, Multiphysics Applications
Eric C. Cyr, Computational Mathematics

February 9, 2016

Abstract

FY16 Key Deliverables – Time Integration (Q4): Deliver an initial time integration API that includes support for IMEX and adjoint sensitivity analysis. Implement basic time integration methods needed to support ATDM Applications. Demonstrate temporal order of accuracy on basic physics test problems representative of the L2 FY16 milestone on ASC testbeds.

Acknowledgment

The authors would like to thanks all those that contributed to the collection of requirements for Tempus: Joe Castro, Larry Musson, Eric Phipps, Bill Rider, Micah Howard, Matt Bettencourt, Allen Robinson, John Shadid, Andy Salinger, (yet to be interviewed: David Littlewood, Stephan Domino, Steve Bova, Travis Fisher, Kendall Pierson, Garth Reese, Eric Keiter, Ting Mei, Shawn Paytz, Bob Schmitt).

Contents

1 Basic Time-Integration Requirements

The following is a list of basic requirements needed by most time integrators.

1. General requirements
   (a) Easy to use! Keep it very simple!
      i. Simple integrators should be simple to do! Complex integrators may (and usually do) require complex methods.
2. Input specification
   (a) Simple to setup time integrators.
   (b) Keep number of lists and nesting to a minimum.
   (c) Common parameters should move up the hierarchy.
   (d) Separate list for time-integrator construction and runtime execution.
3. Output specification
   (a) Have a coherent method for verbosity specification (too easy to get too much information).
   (b) Ensure default output is not too verbose.
   (c) Needs simple clear organization.
4. Software design (General)
   (a) Keep a simple object hierarchy.
   (b) Modular setup and usage.
   (c) Do not use too many levels of indirection.
   (d) Use a simple naming convention.
   (e) Make usage of methods clean and straight forward. (Usage of implicit methods is clunky.)
   (f) Time integrators and steppers need to be well documented
      i. Capabilities and properties of each integrator and stepper needs to be documented.
      ii. Stability, order, ...
      iii. What is functional and what is not? Ramping, error control, ...
      iv. Examples demonstrating capabilities.
         A. If a capability is not regression tested, it does not exist.
   (g) Needs to be built with
      i. Thyra and/or adaptor interface
      ii. ModelEvaluator and/or adaptor interface
      iii. Templated on scalar types
      iv. Will entail using the Data Warehouse?
   (h) Needs to be easy to write a time integrator
      i. Insulate users and application developers from gory details (Thyra, ModelEvaluator) as much as possible.
      ii. Have a pure virtual interface, and a default implementation (stay away from mix-in interfaces i.e., multiple inheritance interfaces)
   (i) Should not incur any additional costs to use the time-integration package.
      i. Obviously minor cost differences should be OK, but with explicit evaluations any additional costs may not be acceptable.

ii. Computational cost of time-integrators are very small.

5. Software design (Specific)
   (a) Construction from ParameterList, solution vector, initial guess, ...
   (b) Easy accessors to objects and parameters
      i. Time step, preferred next time step, time step limits (max/min), ...
      ii. Time-integration errors
   (c) Ability to set/reset functions: ramping, error control, timestep limit functions, ...
   (d) Ability to take a single time step
   (e) Ability to advance the solution to a specific time, and get solutions out at specific times (check pointed solutions not interpolations).
   (f) Ability for step/error control
      i. For a single time step just return error estimates
      ii. For time advance, control step size and error to tolerances and return vector of step sizes taken and error estimates.
      iii. Adjust timestep size based on stability limits (e.g., CFL limits) for each physics.
         A. This will require hooks for the physics to supply timestep limit functions.
      iv. Adjusting timestep size based on temporal errors should be something handled internal to the time integration.
   (g) Ability to ramp the timestep size
   (h) Need access to previous time steps/errors
   (i) Ability to insert observers into the time integration
      i. Need observers between RK stages
   (j) Ability for "out of the box" time integration (i.e., use functionality already available in time-integration package).
      i. For simple time-integration schemes, the user should be able to construct a time integrator and advance a solution "out of the box".
   (k) Ability for "build your own" time integration to construct a specialized time integration scheme
      i. This requires a simple, clear, well-documented definition of a time integrator.
   (l) Forward and adjoint sensitivities.
      i. Should be able to use Griewank's and Wang's method for checkpointing
      ii. Should be able to use reduced memory checkpointing.
      iii. Use Multi-Vector?
      iv. Get sensitivities of solution and response with respect to parameters.
      v. Need an adjoint solve.

2 ATDM Specific Time-Integration Requirements

1. Build the time-integrator package from the ATDM applications, and generalize from there.
2. Time-integration package needs to handle
   (a) Multi-physics, Multi-scale
   (b) Inter-node Asynchronous Multi-Task (AMT)

i. New physics startup (ex: ablation needing new physics, PIC particle hitting boundary creating new particle/physics?)
   - Does this mean just terms in the equations and/or new equations?

ii. New work distributions

3. Steppers

   (a) First-order PDEs

      i. Forward and Backward Euler

      ii. Runge-Kutta

      iii. Implicit and Explicit (IMEX) schemes

         A. Currently working through Piro, but has some issues. (EM)

         B. Need to be able to solve for explicit variables, and eliminate them from the implicit solve.

         C. Need to be able to handle the cases of Lagrangian and ALE formulation (mass matrix).

   (b) Operator-Splitting

      i. First-Order split

      ii. Strang/Marchuk splitting

      iii. Subcycling?

      iv. Can re-solve the current time step

         A. In multi-physics black-box mode coupling,

            - User queries each physics for preferred timestep size, and timestep limits.
            - Next timestep is negotiated between the preferred timestep sizes.
            - Each physics takes the negotiated timestep size.
            - If any of the physics fails the timestep, tell ALL physics to reject the current solution and re-solve using a new negotiated (smaller) timestep.
            - The PIKE black-box ModelEvaluator is an example of this.

   (c) L-stable methods

   (d) SSP methods

   (e) Second-order PDEs

      i. Trapezoidal Rule

      ii. Newmark-β

      iii. Generalized-α Method (Emphasis, Charon 2)

      iv. Leap-Frog, Velocity-Verlet (PIC)

   (f) BDF (predictor-corrector)

      i. For multi-step time integrators, ability to reset the history and restart with new time integrator

         A. Solve to steady state, throw away history, restart with new time integrator

4. Need to be able to re-calculate the state/residual from scratch, and when only part of the state has changed.

   (a) Example: in a Lagrangian calculation, you might need to recalculate but the mesh coordinates have not changed.

3 Additional Time-Integration Requirements

1. Ability to switch between time integrators
2. Ability to input user-defined Butcher Tableaus
3. Ability to do Matrix-Free implicit methods
4. Uncertainty Quantification
   (a) Embedded Ensemble Propagation
   (b) Embedded Stochastic Galerkin (SG) - Polynomial chaos approximation
5. Get the time derivatives of auxiliary variables that are not part of the solution vector
   (a) Analytic (chain rule)
      i. (+) Provides a conservative (as in conservation of energy) value.
         A. Maintain divergence-free properties.
      ii. (-) As the number of dependent variables of the auxiliary variable changes, you need to recode the chain rule.
   (b) Finite-difference/polynomial interpolation
      i. (-) Does not necessarily provide a conservative (as in conversation of energy) value.
      ii. (+) Insensitive to the number of dependent variables.
      iii. (-) Need a stateless function evaluation for the auxiliary variable.
   (c) Can we use the DAE index "evaluation" to get the time derivative of the auxiliary "algebraic" variable? This is probably very similar to the analytic and chain rule method.
   (d) This should include time integrated values (e.g., some adjoint and optimization quantities).
6. Steppers
   (a) Explicit time integrator
      i. Central difference (transient dynamics, CompSim User's Manual Salinas)
7. Support Hessian information for forward sensitivities (i.e., second derivatives wrt parameters).
8. Support Space-Time formulation
9. Support for full-space optimization methods
   (a) Full KKT system
10. Provide time-integration interface for third-party libraries
    (a) This could provide comparisons for verification and performance assessment.
    (b) SUNDIALS (Carol Woodward, LLNL, and Dan Reynolds, SMU) is a candidate for this.

4 Interoperability

Other packages that the time integrator needs to interact with.

4.1 Thyra

4.2 ModelEvaluator

1. Motivation for MEs
   (a) Provide single interface from nonlinear ANAs (NOX, Rythmos, LOCA, MOOCHO) to applications (for numerical algorithms. Still need a "usage" interface to each ANA.).
   (b) Provides shared, uniform access to linear solver capabilities
   (c) Once an application implements support for one ANA, support for other ANAs can quickly follow (Incremental support for sensitivities, optimization, and UQ)
   (d) Mixed problem types will become more and more common and must be easy to support
      i. Transient optimization
      ii. Uncertainty in transient simulations
      iii. Stability of an optimum under uncertainty of a transient problem
2. MEs are stateless!
   (a) State values are passed in as a parameter.
      i. This requires complete recompute of the ME.
      ii. What if some compute extensive pieces of the state do not need to be recomputed? Wasted cycles!
      iii. Can we have a "stateful" ME, which could keep track the of state through the InArgs?
3. InArgs and OutArgs
   (a) Are confusing - "not intuitive"
   (b) Eric Cyr proposed alternative methods using templating?
4. Need MEs for time integration
   (a) This will inherently cause any new design of the numerical algorithms to look similar to Rythmos.
   (b) Can we improve the situation?
      i. Refactor of the current ME classes.
      ii. Create an adapter from the ME to the new time integrator.
         - Some packages are already doing this (Piro (Opt), Drekar (SimOpt), ROL, ...)
         - Abstract operator and vector
5. Material Models have state!
   (a) Linear Visco-elastic
   (b) Joints and hysteresis.

4.3 PIRO

1. PIRO (Parameters In, Response Out)
   (a) Collection of steps of the common usage for NOX, Rythmos, LOCA, ...
   (b) Takes in a ME and ParameterList
      i. Constructor of Rythmos time integrator
      ii. Passes ParameterList to Rythmos – Solve

iii. Sensitivity Analysis

(c) ROME for ROL and Dakota

(d) PIRO has functioning second-order PDE time integrator

i. Methods

A. Newmark-β

B. Trapezoidal rule?

C. Velocity-Verlet

ii. Constant time step

iii. No sensitivity analysis

iv. EpetraEXT versions only (Ross is putting in Thyra capabilities shortly?)

v. Material model state through observe_solution

vi. Can integrate energy though may still have some bugs.

vii. Ability to invert mass matrix for explicit methods (IMMD - Invert Mass Matrix Decorator)

$$\begin{aligned} \text{Albany sees} \quad M\dot{x} &= F(x) \\ \text{Rythmos sees} \quad \dot{x} &= M^{-1}F(x) \end{aligned}$$

viii. Ability to lump

4.4 Sacado