SAND2016-10775PE

Tempus Time-Integration Package

Curtis Ober

Roger Pawlowski

Eric Cyr

Sandia National Laboratories

Exceptional service in the national interest

Trilinos User-Developer Group Meeting
1:50-2:10pm October 25, 2016

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

What is Tempus?

CENTER FOR COMPUTING RESEARCH

Sandia National Laboratories

Tempu upervillan on the te show
Lois & The New Adventures rman

What is Tempus?

- Latin for time
  - "tempus fugit" → "time flies"
- New time-integration package
  - Replacement for Rythmos
  - Make improvements to the design and interface
  - Scavenge what we can from Rythmos
- Gathered Requirements
  - Interviewed ~12 people w/ and w/o experience with Rythmos
  - Tried to determine what worked and what did not
  - Received good advice and contradictory advice
    - e.g., "Use Thyra" and "Don't use Thyra"
  - Biggest "nugget" I learned
    - We will not be able to fix all the complaints about Rythmos, because some are related to ModelEvaluators and Thyra.

General Requirements

- Keep it simple and easy to use.
- Have well documented examples and usage
- Provide time integrators for “out-of-the-box” usage
- Provide components to “build-your-own” time integration
- Work with other Trilinos packages (Algebraic Numerical Algorithms)
- Basic capabilities need to include
  - ODEs, DAE’s, forward and adjoint sensitivities
  - First- and second-order PDE integration
  - Single and multi-physics time integration
- Access to the time integration for application functions
- Time-integration computational and memory costs should be kept to a minimum

ATDM Requirements

- FY16 Key Deliverables
  - 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.

- FY 17 Key deliverable
  - Develop and demonstrate an IMEX scheme in Tempus to support EMPIRE fluid solver.
  - Demonstrate IMEX solver and document performance on problem of interest for EMPIRE.

Fully Implicit DAE/ODE

$$\mathbf{f}(\ddot{\mathbf{x}}(t), \dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{p}, t) = 0 \qquad \text{for } t \in [t_0, t_f]$$

$$\mathbf{x}(t_0) = \mathbf{x}_0(\mathbf{p})$$

State ⟶ $\dot{\mathbf{x}}(t_0) = \dot{\mathbf{x}}_0(\mathbf{p})$

$$\ddot{\mathbf{x}}(t_0) = \ddot{\mathbf{x}}_0(\mathbf{p})$$

$x \in \mathcal{X}$ is the vector of differential state variables,

$\dot{x} \in \mathcal{X}$ is the vector of temporal derivatives of x,

$\ddot{x} \in \mathcal{X}$ is the vector of temporal second derivatives of x,

$t, t_0, t_f \in \mathbb{R}$ are the current, initial, and the final times respectively,

$f(\ddot{x}, \dot{x}, x, t) \in \mathcal{X}^2 \times \mathbb{R} \to \mathcal{F}$ defines the DAE vector function,

$\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the vector space of the state variables x, and

$\mathcal{F} \subseteq \mathbb{R}^{n_x}$ is the vector space of the output of the DAE function $f(\ldots)$.

- Advance the state to final time ⟶ Integrator
- Achieved through a sequence of smaller time steps ⟶ Stepper

Tempus::SolutionState

- Primary Design Consideration
  - Encapsulate the state of the solution
  - Should able to restart the integration from a SolutionState
    - Needed for check-pointing and “undo”
- SolutionState contains
  - State – $\mathbf{x}(t), \dot{\mathbf{x}}(t), \ddot{\mathbf{x}}(t)$
  - MetaData – time, index, order, error, restartable, interpolated, …
  - StepperState – data that the stepper needs to restart
  - PhysicsState – any data needed for the physics

managed by Tempus

provided by Application

Tempus::SolutionHistory

- Storage mechanism for the solution history
  - A container of SolutionStates
  - Chronological and index access
  - Manages various access patterns
    - Adjoint sensitivities require storage of forward solution
      - Need check-pointing capabilities, e.g., Griewank
    - Some time integrators need past time steps, e.g., BDF methods
- Provide interpolation capabilities
  - Useful for adjoint sensitivities, e.g., stages of Runge-Kutta
  - Warning – interpolated solutions do not likely satisfy conservation!
  - Scavenge from Rythmos::InterpolationBuffer
- Possibly use Data Warehouse?
  - Storage can be in a variety locations, e.g.,
    - In-core, on disk, on-processor/GPU and mixed capabilities

Integrator/Stepper

- Fully Implicit DAE/ODE

$$\mathbf{f}(\ddot{\mathbf{x}}(t), \dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{p}, t) = 0 \qquad \text{for } t \in [t_i, t_{i+1}] \quad \text{for } i = 0, \ldots, N-1$$

$$\mathbf{x}(t_i) = \mathbf{x}_i(\mathbf{p})$$
$$\dot{\mathbf{x}}(t_i) = \dot{\mathbf{x}}_i(\mathbf{p})$$
$$\ddot{\mathbf{x}}(t_i) = \ddot{\mathbf{x}}_i(\mathbf{p})$$

- Steppers
  - Take a **single** time step (PASS/FAIL)
  - Steppers require
    - ModelEvaluator(s),
    - SolutionState(s), and/or
    - Solver(s)
  - Can suggested dt
  - Can have sub-Steppers

- Integrators
  - Is the time "loop"
  - Has a single Stepper
  - Determines time step size
  - Output results
  - Holds the SolutionHistory
  - Does not call ModelEvaluator
  - Interact w/App thru Observers

Tempus Integrators

- Tempus::IntegratorSimple
  - Just a simple time “loop”
- Tempus::IntegratorBasic
  - Demonstrates all the basic capabilities
  - Has SolutionHistory for Stepper, undo and adjoint sensitivities
  - Has Observers to perform application-defined functions
  - Has TimeStepControl for
    - Basic bounding of time step
    - Hitting output times

Tempus Steppers

- Forward Euler
- Backward Euler
- Explicit Runge-Kutta (ERK)
  - 10 Butcher Tableaus
- Diagonally Implicit Runge-Kutta (DIRK)
  - 5 Butcher Tableaus
- IMEX for EMPIRE
- RK and BDF2 for SPARC
- …

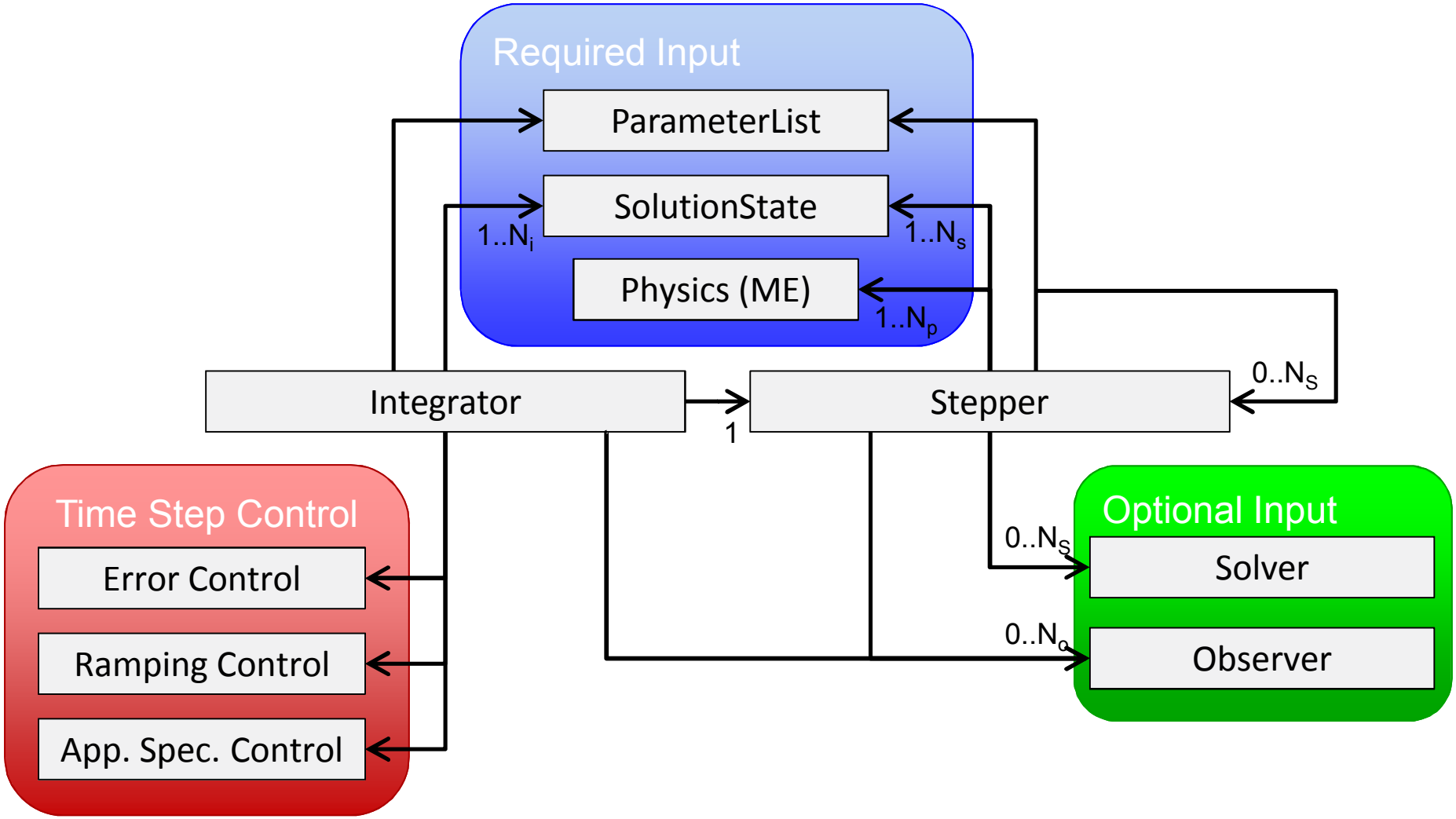
Tempus::TimeStepControl

- Determine the time step for the Integrator
- Base class has basic capabilities
  - Simulation time min/max bounding
  - Time index min/max bounding
  - Time step size min/max bounding
  - Relative/Absolute maximum error
  - Order min/max bounding
  - Time-step adjustments for output
  - Maximum number of failures and consecutive failures
  - Ensure constant time steps
  - Incorporate Stepper suggested time step
- Derived classes for additional controls, e.g., ramping
- Also have application specific time-step control through Observer

Observer

- Observers are a means to “inject” app-defined functions within a process.
- Integrators and Steppers will have observers after every major component, e.g.,
  - Integrators
    - observeStartIntegrator()
    - observeStartTimeStep()
    - observeNextTimeStep()
    - observeBeforeTimeStep()
    - observeAfterTimeStep()
    - observeAcceptTimeStep()
    - observeEndIntegrator()
  - Stepper
    - observeStartStepper()
    - …
    - observeAcceptStep()
    - observeEndStepper()
- If Observer is not sufficient, application can “build their own” Integrator or Stepper.

Interface Design



Example Tempus::StepperForwardEuler

CENTER FOR COMPUTING RESEARCH

Sandia National Laboratories

ModelEvaluator::SinCosModel (scavenged from Rythmos)

Governing Equation

$$\ddot{x} = -x$$

Reduced first-order system

$$\frac{d}{dt}x_0(t) = x_1(t)$$

$$\frac{d}{dt}x_1(t) = \left(\frac{f}{L}\right)^2 (a - x_0(t))$$

SinCosModel ->
- Implicit model formulation = 0
- Accept model parameters = 0
- Provide nominal values = 1
- Coeff a = 0
- Coeff f = 1
- Coeff L = 1
- IC x_0 = 0
- IC x_1 = 1
- IC t_0 = 0

Initial Conditions

$$x_0(t_0 = 0) = \gamma_0[= 0]$$

$$x_1(t_0 = 0) = \gamma_1[= 1]$$

Parameters

$$a = 0 \quad f = 1 \quad L = 1$$

Exact Solution

$$x_0(t) = a + b * \sin((f/L) * t + \phi)$$

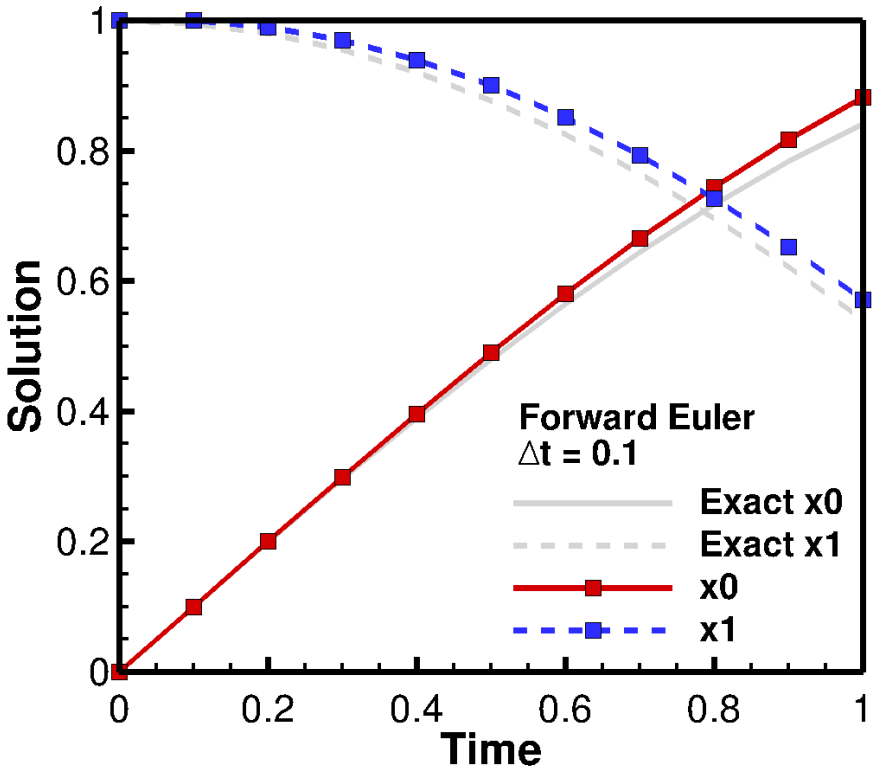
$$x_1(t) = b * (f/L) * \cos((f/L) * t + \phi)$$

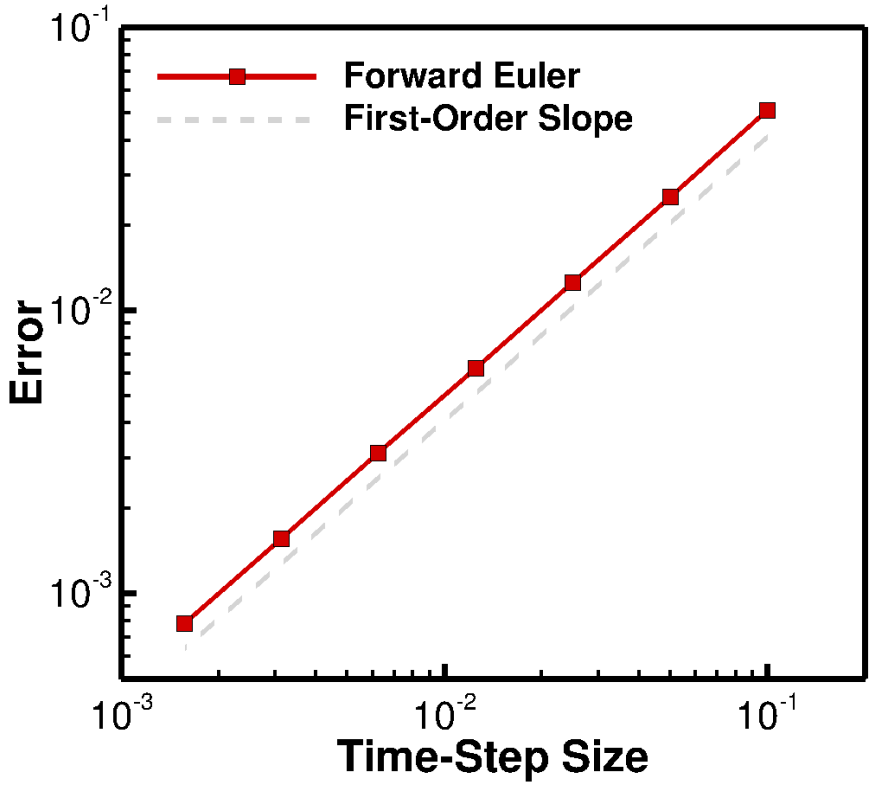
$$\phi = \arctan(((f/L)/\gamma_1) * (\gamma_0 - a)) - (f/L) * t_0[= 0]$$

$$b = \gamma_1/((f/L) * cos((f/L) * t_0 + \phi))[= 1]$$

Example Tempus::StepperForwardEuler

ModelEvaluator::SinCosModel (scavenged from Rythmos)

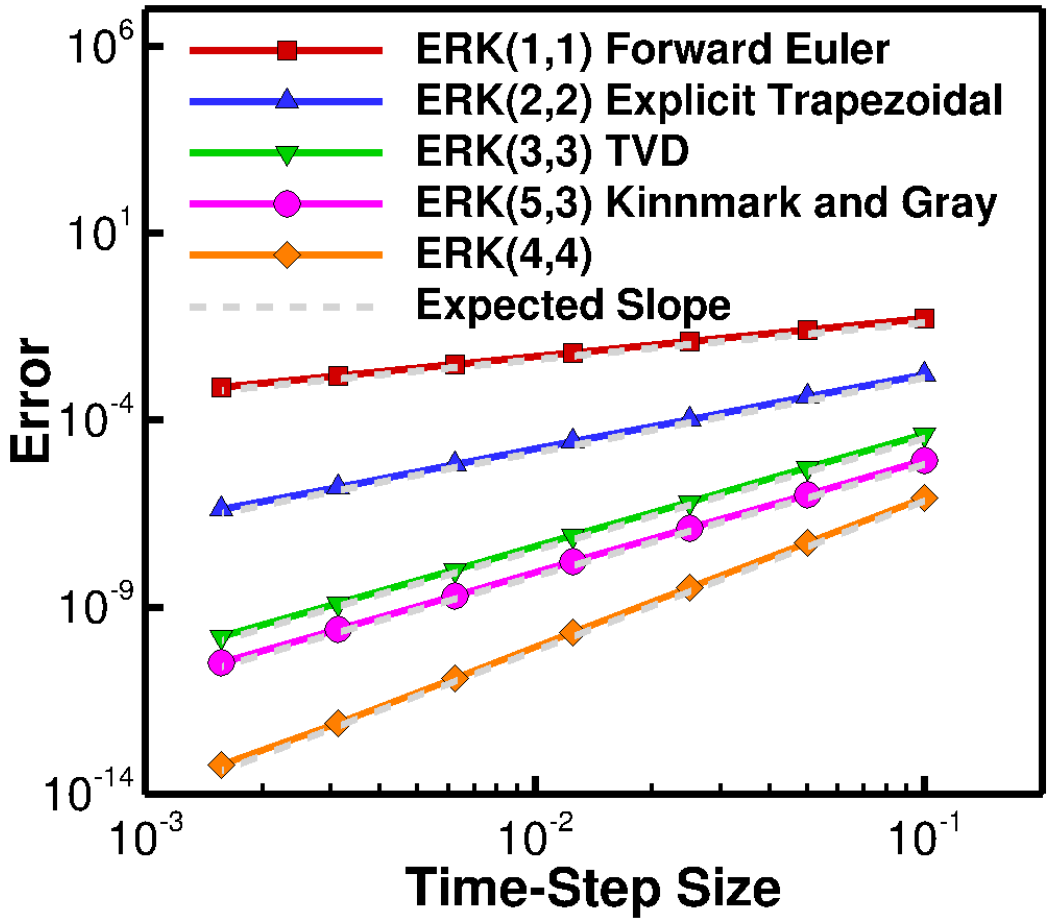




Example
Tempus::StepperExplicitRK

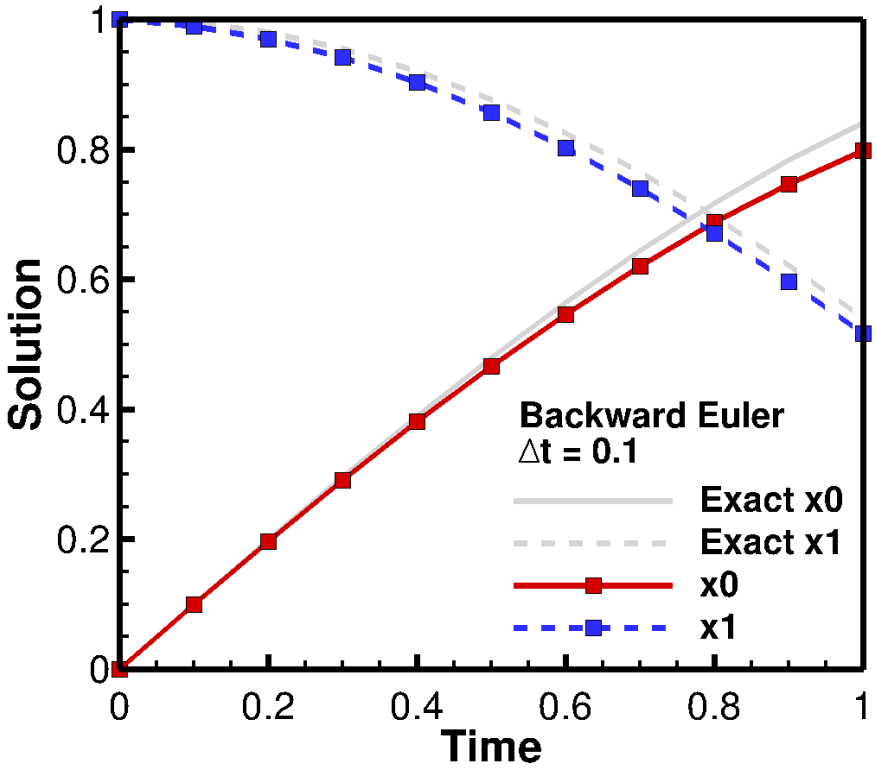
CENTER FOR COMPUTING RESEARCH
Sandia National Laboratories

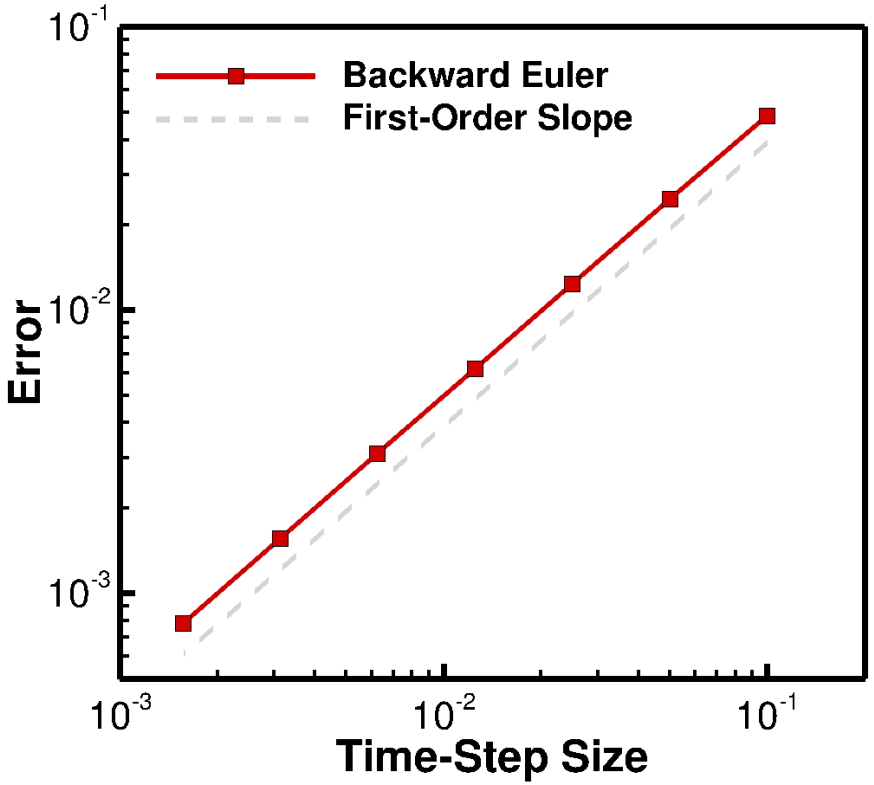
ModelEvaluator::SinCosModel (scavenged from Rythmos)



Example
Tempus::StepperBackwardEuler

ModelEvaluator::SinCosModel (scavenged from Rythmos)





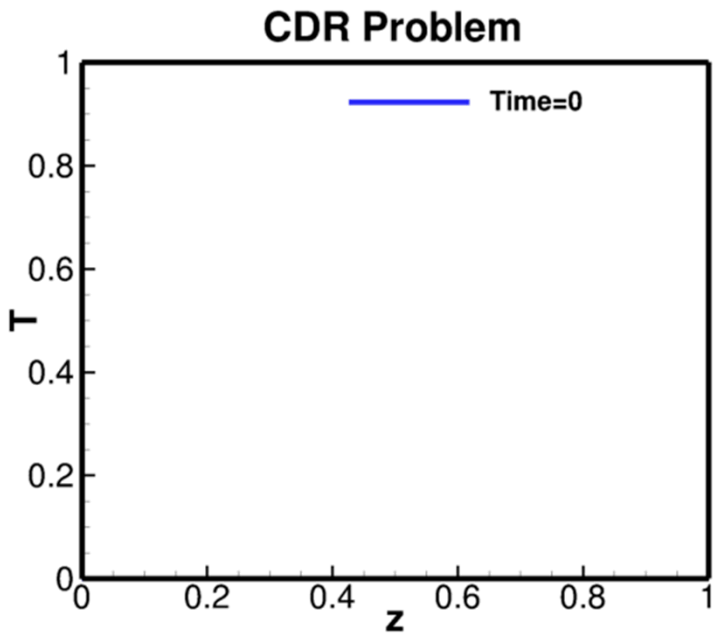
CDR Problem

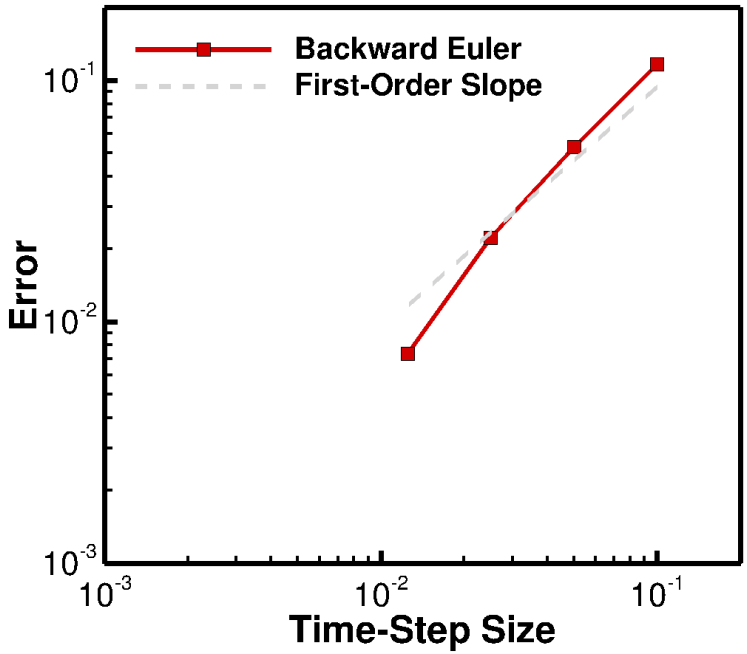
- Convection-Diffusion-Reaction

$$\frac{\partial T}{\partial t} + a\frac{\partial T}{\partial z} + \frac{\partial^2 T}{\partial z^2} - K\,T^2 = 0 \qquad \text{for } t \in [0,1] \text{ and } z \in [0,1]$$

$$\text{with ICs} \quad T(z,0) = 0 \qquad \text{with BCs} \quad T(0,t) = 1 \qquad \text{where} \quad a = 0.1$$

$$\frac{\partial T}{\partial t}(z,0) = 0 \qquad \frac{\partial T}{\partial z}(1,t) = 0 \qquad K = 1$$





Summary

- Developed new design for time integration
  - Improved interface and design for usability
  - Documented design and tests for new developers and users
  - Provide “out-of-the-box” and “build-your-own” capabilities
- Current capabilities
  - Forward and Backward Euler
  - Explicit Runge-Kutta
  - Implicit Runge-Kutta
- Currently accessible via stand-alone repo
  - FY2017 should become a package in Trilinos
- Capabilities in FY2017
  - IMEX methods
  - Forward and Adjoint Sensitivities