SAND2008-7486P

Rythmos

Addressing the error of our ways

Trilinos Users Group

October 21, 2008

Todd Coffey
Computational Mathematics & Applications
Roscoe Bartlett
Uncertainty Quantification & Optimization

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Addressing the Error of our Ways

- Global Error Estimation
- Runge-Kutta Updates
- Adjoint Integration
- Path Forward
- Conclusion

Global Error Estimation

- What is this?
- A-posteriori goal-oriented error estimation
  - End point error
  - Average error
- Don Estep / Jeff Sandelin collaboration (CSU)
- GAASP code
- Rythmos/GAASP coupling

Adjoint System

- **Consider the ODE:**

$$\begin{cases} \dot{y} &= f(y,t) \\ y(0) &= y_0 \end{cases}$$

- **And the adjoint system:**

$$\begin{cases} -\dot{\varphi} &= \hat{f}'(Y(t),t)^{\top}\varphi \\ \varphi(T) &= \psi \end{cases}$$

- **Where:**

$$\hat{f}'v = \int_0^1 f'(sy + (1-s)Y)ds\, v$$

- **(Dis)Continuous Galerkin in time resulting in IRK**

Global Error Estimate

$$(e(T), \psi) =$$

$$(e(0), \varphi(0))$$

$$-\sum_{n=1}^{N} \int_{t_{n-1}}^{t_n} \left(\dot{Y} - f(Y,t)\right) (\varphi - \pi_k \varphi)\, dt$$

$$-\sum_{n=1}^{N} \left([Y_{n-1}], (\varphi - \pi_k \varphi)^+_{n-1} \right)$$

} **Discretization Error**

$$-\sum_{n=1}^{N} \left(\int_{t_{n-1}}^{t_n} (f(Y,t), \varphi)\, dt - \int_{t_{n-1}}^{t_n} \left(\overline{f(Y,t)}, \varphi\right) dt \right)$$

} **Quadrature Error**

Sandia National Laboratories

Global Error Flow

- So, how does this work?
- Forward solve to compute: $Y(t), \quad t = 0 \ldots T$
- Adjoint solve to compute: $\varphi(t), \quad t = T \ldots 0$
- Compute error estimate: $(e(T), \psi)$
- Refine step-sizes to control global error

GAASP code

- Globally Accurate Adaptive Sensitivity Package
- dG/cG methods
- Error estimation breakdown
- Model requirements: diffjac
- Error control algorithm
- Cubic spline smoothing
- Quadrature

Rythmos/GAASP coupling

- Model evaluator interface
- Supply residual loads

Rythmos GEE Migration

- dG/cG IRK methods
- Cubic spline interpolation buffer
- Quadrature
- Adjoint integration
- Error estimation object
- Error control strategies

Implicit Runge-Kutta Method

- **Differential-Algebraic Equation:** $F(t, y, \dot{y}) = 0$
- **RK method solves the system:**

$$F_i = F(Y_i^{'}, y_{n-1} + h \sum_{j=1}^{s} a_{ij} Y_i^{'}, t_{n-1} + c_i h), \quad i = 1 \ldots s$$

- **RK solution:** $y_n = y_{n-1} + h \sum_{i=1}^{s} b_i Y_i^{'}$

- **RK Butcher Tableau:**

c_1	a_{11}	a_{12}	a_{13}
c_2	a_{21}	a_{22}	a_{23}
c_3	a_{31}	a_{32}	a_{33}
	b_1	b_2	b_3

Runge-Kutta Butcher Tableau

- Defines combination of A, b, and c
- Accepts description and order
- Factory will create one by name, order, or type
- Once registered with factory, it will be convergence tested automatically
- Can assert: Explicit, Implicit, Diagonally implicit, and singly diagonally implicit tableaus
- Any RK tableau can be passed into both the ERK and the IRK steppers.

Implicit RK Tableaus

- **Implicit RK Butcher Tableau:**

c_1	a_{11}	a_{12}	a_{13}
c_2	a_{21}	a_{22}	a_{23}
c_3	a_{31}	a_{32}	a_{33}
	b_1	b_2	b_3

- **Diagonally Implicit RK:**

c_1	a_{11}	0	0
c_2	a_{21}	a_{22}	0
c_3	a_{31}	a_{32}	a_{33}
	b_1	b_2	b_3

- **Singly Diagonally Implicit RK:**

c_1	γ	0	0
c_2	a_{21}	γ	0
c_3	a_{31}	a_{32}	γ
	b_1	b_2	b_3

Implicit RK Stepper

- Unit testing
- Convergence testing
- (S)DIRK specialization

IRK Implementation

Thyra Product Vector Use:

```
// Rythmos IRK Model Evaluator Code Snippet…
 for ( int i = 0; i < numStages; ++i ) {
    // B.1) Setup the DAE's inArgs for stage f(i) ...
    assembleIRKState( i, irkButcherTableau_.A(), delta_t_, *x_old_, *x_bar,
   outArg(*x_i) );
    daeInArgs.set_x( x_i );
    daeInArgs.set_x_dot( x_bar->getVectorBlock(i) );
    daeInArgs.set_t( t_old_ + irkButcherTableau_.c()(i) * delta_t_ );

    // B.2) Setup the DAE's outArgs for stage f(i) ...
    if (!is_null(f_bar))
      daeOutArgs.set_f( f_bar->getNonconstVectorBlock(i) );
   }
}
```

Sandia National Laboratories

More IRK Implementation

Thyra composite operators

```
// Rythmos IRK Model Evaluator Code Snippet…

if (!is_null(W_op_bar)) {
      for ( int j = 1; j < numStages; ++j ) {
        Scalar alpha = ST::zero();
        if (i == j) {
          alpha = ST::one();
        } else {
          alpha = ST::zero();
        }
        Scalar beta = delta_t_ * irkButcherTableau_.A()(i,j);
        daeInArgs.set_alpha( alpha );
        daeInArgs.set_beta( beta );
        daeOutArgs.set_W_op(W_op_bar->getNonconstBlock(i,j));
        daeModel_->evalModel( daeInArgs, daeOutArgs );
        daeOutArgs.set_W_op(Teuchos::null);
      }
    }
```

IRK Stepper Usage

- Same requirements as ImplicitBDFStepper
  - Nonlinear Solver
  - Jacobian
- Runge-Kutta Butcher Tableau

Explicit RK Method

- **ODE:**

$$\begin{cases} \dot{y} &= f(y,t) \\ y(0) &= y_0 \end{cases}$$

- **ERK method:**

$$Y_i^{'} = f(y_{n-1} + \sum_{j=1}^{i-1} a_{ij} Y_j^{'}, t_{n-1} + c_i h), \quad i = 1 \ldots s$$

- **ERK Solution:**

$$y_n = y_{n-1} + h \sum_{i=1}^{s} b_i Y_i^{'}$$

- **Butcher Tableau:**

c_1	0	0	0
c_2	a_{21}	0	0
c_3	a_{31}	a_{32}	0
	b_1	b_2	b_3

ERK Update

- Previously only supported 4 stage 4^{th} order ERK method
- Now accept any explicit RK Butcher Tableau
- Full convergence tests
- Default constructor assumes 4 stage ERK method

Convergence Testing

- Sine/Cosine problem: $\ddot{y} = -y$
- Diagonal problem: $\dot{y} = \lambda y$
- Compute solution to fixed time using finer meshes
- Compare error to exact solution
- Compute least squares linear fit to log plot
- Slope of linear fit is order of method

Unit Testing

- **See my Test Driven Development talk on Thursday**
- **Teuchos Unit Test Harness (Ross)**
- **Example:**

```
TEUCHOS_UNIT_TEST( Rythmos_RKButcherTableau,
   createBackwardEuler_RKBT ) {
  RKButcherTableau<double> rkbt = createBackwardEuler_RKBT<double>();
  TEST_EQUALITY_CONST( rkbt.numStages(), 1 );
  TEST_EQUALITY_CONST( rkbt.A()(0,0), 1.0 );
  TEST_EQUALITY_CONST( rkbt.b()(0), 1.0 );
  TEST_EQUALITY_CONST( rkbt.c()(0), 1.0 );
  TEST_EQUALITY_CONST( rkbt.order(), 1 );
}
```

Global Error Path

- Have IRK, need adjoints
- Along the way:
  - Trilinos Vertical Integration Milestone (Charon)
  - Aria Adjoint integration work

Vertical Integration Milestone

- Rythmos improvements to support Charon:
  - BDF step-size strategy
  - BDF error-weight norm strategy
  - Integrator implementation
  - Integrator step control strategy
  - Observers
  - Forward Sensitivities
  - Breakpoint support
  - Numerous bugfixes & other improvements

Adjoint Integration

- Implemented to support adjoints in SIERRA/Aria
- New Functionality in Rythmos
- Supports: Linear Adjoint integration with end-point specification
- Model Evaluator implementation
- Tested in Aria nightly

Efficiency…

- How can we reduce the cost of adjoint based global error estimation?
  - Pre-compute error estimates for model problem
  - Post-compute error estimates in down-time
  - Parallel time integration to hide cost

Research Front

- Global error estimates for DAEs
- Global error estimates for BDF
- Relate model problem error estimates to specific problem
- Global error control strategies
- Parallel time integration based on adjoint error estimates

Path Forward

- To complete Global Error Estimation Algorithms:
  - Cubic Spline InterpolationBuffer
  - Quadrature algorithms
  - Error Estimation algorithms from GAASP
- To continue application integration:
  - More sophisticated adjoint integration
  - Restarts and checkpointing support
  - Variable step-size algorithms for RK methods
  - Continue expanding & refining: Charon & Aria support
  - Develop Xyce integration support

Until next time…

Rythmos

Addressing the error of our ways

Global Error Estimates

Unit Tests

Convergence Tests

Thank you!

Sandia National Laboratories