



Rythmos

Addressing the error of our ways

Trilinos Users Group
October 21, 2008

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



What Is Rythmos?

- **Transient Differential Equation Solver**

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

- **DAEs:**
$$\begin{cases} F(\dot{y}, y, t) &= 0 \\ y(0) &= y_0 \\ \dot{y}(0) &= \dot{y}_0 \end{cases}$$

- **Algorithms:** implicit BDF, implicit/explicit RK, integrators, forward/adjoint sensitivities, more...



Addressing the Error of our Ways

- **Global Error Estimation**
- **Runge-Kutta Updates**
- **Adjoint Integration**
- **Research**
- **Path Forward**



Global Error Estimation

- What is this?
- Goal oriented a posteriori error estimation
 - End point error: $e(T) = y(T) - Y(T)$
 - Average error: $\frac{1}{N} \sum_{n=1}^N \|e(t_n)\|$
- Donald Estep / Jeff Sandelin collaboration (CSU)
- GAASP code
- Rythmos/GAASP coupling



Global Error Estimation

- 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:

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

- Variational formulation

- (Dis)Continuous Galerkin in time resulting in IRK



Global Error Estimate

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

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

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

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

Discretization Error

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

Quadrature Error



Global Error Process

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



GAASP Code

- **G**lobally **A**ccurate **A**daptive **S**ensitivity **P**ackage
- PhD research code written by Jeff Sandelin, CSU
- dG/cG methods (resulting in IRK)
- Error estimation with/without cancellation
- Model requirements: finite difference jacobians
- Error control algorithms
- Cubic spline smoothing of forward solution
- Quadrature approximations for integrals



Rythmos/GAASP coupling

- **Model evaluator interface**
- **Residual only loads (at this point)**
- **GAASPErrorEstimator class**
 - **setQuantityOfInterest**
 - **getErrorEstimate**
 - **controlGlobalError**
- **GAASPErrorEstimate class**
 - **getTotalError**



GAASP Algorithms -> Rythmos

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



Implicit Runge-Kutta Method

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

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

- **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**
- **Can assert: explicit, implicit, diagonally implicit, and singly diagonally implicit tableaus**
- **Once registered with factory, it will be convergence tested automatically**

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



IRK Implementation

Thyra Product Vector Use:

```
typedef Thyra::ProductVectorBase<Scalar> PVB;
const RCP<const PVB> x_bar = rcp_dynamic_cast<const PVB>(inArgs_bar.get_x(),
    true);
const RCP<PVB> f_bar = rcp_dynamic_cast<PVB>(outArgs_bar.get_f(), true);
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) ...
    daeOutArgs.set_f( f_bar->getNonconstVectorBlock(i) );
    // B.3) Compute f_bar(i)
    daeModel_->evalModel( daeInArgs, daeOutArgs );
    daeOutArgs.set_f(Teuchos::null);
}
```



More IRK Implementation

Thyra composite operators

```
typedef Thyra::BlockedLinearOpBase<Scalar> BLWB;
const RCP<BLWB> W_op_bar = rcp_dynamic_cast<BLWB>(outArgs_bar.get_W_op(), true);
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

- **Similar requirements as ImplicitBDFStepper**
 - Nonlinear Solver
 - Jacobian
- **Runge-Kutta Butcher Tableau**
- **Enlarged matrix solver**

```
stateStepper = Rythmos::implicitRKStepper<double>(
    stateModel, nonlinearSolver, irk_W_factory, irkbt
);
double dt = 1.0;
double dt_taken = stateStepper.takeStep(dt,Rythmos::STEP_TYPE_FIXED);
```




Explicit RK Method

- ODE:
$$\begin{cases} \dot{y} = f(y, t) \\ y(0) = y_0 \end{cases}$$
- ERK method:

$$Y'_i = f\left(y_{n-1} + \sum_{j=1}^{i-1} a_{ij} Y'_j, t_{n-1} + c_i h\right), \quad i = 1 \dots s$$

- 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 4th 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: $\dot{y} = \lambda y$
- Integrate solution out to fixed time using successively 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

- Along the way:
 - Trilinos Vertical Integration Milestone (Charon)
 - Aria Adjoint integration work
- Have IRK
- **Need Adjoints**
- Cubic spline interpolation buffer
- Quadrature calculations
- Error estimation objects
- Error control strategies



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

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



On our Way...

- Have IRK
- Have Adjoint
- **Whoa! How much does this cost?**
- Cubic spline interpolation buffer
- Quadrature calculations
- Error estimation objects
- Error control strategies



Cost Hiding...

- **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**
- **We betting this is well worth the retail price**



Research Front

- **Global error estimates for DAEs**
- **Global error estimates for BDF methods**
- **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

Implicit RK

Convergence Tests

Thank you!