Multirate time integration with ARKode/SUNDIALS

Daniel R. Reynolds[1], David J. Gardner[2], Rujeko Chinomona[1]

reynolds@smu.edu, gardner48@llnl.gov, rchinomona@smu.edu

(significant contributions from C.J. Balos[2], J. Loffeld[2], and C.S. Woodward[2])

[1] Department of Mathematics, Southern Methodist University
[2] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory

International Congress on Industrial and Applied Mathematics
Valencia, Spain
18 July 2019

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC.

Multiphysics Problems

"Multiphysics" problems typically involve a variety of interacting processes:

- System of components coupled in the bulk [cosmology, combustion]
- System of components coupled across interfaces [climate, tokamak fusion]

Multiphysics simulation challenges include:

- Multirate processes, but too close to analytically reformulate.
- Optimal solvers may exist for some pieces, but not for the whole.
- Mixing of stiff/nonstiff processes, a challenge for standard algorithms.

Historical approaches rely on lowest-order time step splittings, may suffer from:

- Low accuracy – typically $\mathcal{O}(h)$-accurate; symmetrization/extrapolation may improve this but at significant cost [Ropp, Shadid & Ober 2005].
- Poor/unknown stability – even when each part utilizes a 'stable' step size, the combined problem may admit unstable modes [Estep et al., 2007].

Need for Flexible Time Integration Libraries

Multiphysics time integration needs:

- Stability/accuracy for each component, as well as inter-physics couplings
- Custom/flexible step sizes for distinct components
- Robust temporal error estimation & adaptivity of step size(s)
- Built-in support for spatial adaptivity
- Ability to apply optimal solver algorithms for individual components
- Support for testing a variety of methods and solution algorithms

Legacy software frameworks enforce overly-rigid standards on applications:

- Fully implicit or fully explicit, without IMEX flexibility.
- Inflexible data structures for vectors, matrices, (non)linear solvers.
- Hard-coded parameters – good for most problems, but rarely optimal.

SUNDIALS: SUite of Nonlinear and DIfferential-ALgebraic equation Solvers

Software library of ODE and DAE time integrators and nonlinear solvers

- Consists of six packages: CVODE(S), ARKode, IDA(S), and KINSOL
- Written in C with interfaces to Fortran
- Designed to be easily incorporated into existing codes

Modular implementation

- Data use is fully encapsulated by vector and matrix APIs
- Nonlinear and linear solvers are fully encapsulated from the integrators
- All parallelism is encapsulated in vectors, solvers, and user-supplied functions
- Vector, matrix, and solver modules can all be user-supplied

Availability and support

- Freely available; BSD 3-Clause license; $> 25,000$ downloads in 2018
- Detailed user manuals and an active user community email list

More information: `https://computing.llnl.gov/casc/sundials`

- 11:30 Friday: talk by Carol Woodward (MS GH-3-5-9)
- Monday/Tuesday: poster by David Gardner (PA-041)

Additive Runge–Kutta (ARK) Methods [Ascher et al. 1997; Araújo et al. 1997; . . .]

ARKode was initially designed to implement adaptive ARK methods for initial value problems (IVPs), supporting up to two split components: *explicit* and *implicit*,

$$M\dot{y} = f^E(t,y) + f^I(t,y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0,$$

- M is any nonsingular linear operator (mass matrix, typically $M = I$),
- $f^E(t,y)$ contains the explicit terms,
- $f^I(t,y)$ contains the implicit terms.

Combine two s-stage RK methods; denoting $t^*_{n,j} = t_n + c^*_j h_n$, $h_n = t_{n+1} - t_n$:

$$Mz_i = My_n + h_n \sum_{j=1}^{i-1} A^E_{i,j} f^E(t^E_{n,j}, z_j) + h_n \sum_{j=1}^{i} A^I_{i,j} f^I(t^I_{n,j}, z_j), \quad i = 1, \ldots, s,$$

$$My_{n+1} = My_n + h_n \sum_{j=1}^{s} \left[b^E_j f^E(t^E_{n,j}, z_j) + b^I_j f^I(t^I_{n,j}, z_j) \right] \quad \text{(solution)}$$

$$M\tilde{y}_{n+1} = My_n + h_n \sum_{j=1}^{s} \left[\tilde{b}^E_j f^E(t^E_{n,j}, z_j) + \tilde{b}^I_j f^I(t^I_{n,j}, z_j) \right] \quad \text{(embedding)}$$

ARKode Flexibility Enhancements

ARKode includes several enhancements for multiphysics codes, including:

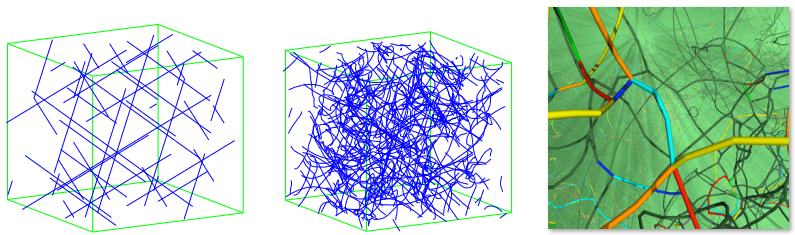
- Variety of built-in Butcher tables (ERK, DIRK, IMEX ARK); supports user-supplied.
- Variety of built-in temporal adaptivity functions; supports user-supplied.
- Variety of built-in implicit predictor algorithms.
- Ability to specify that problem is linearly implicit.
- Ability to resize data structures based on changing IVP size.
- All internal solver parameters are user-modifiable.

Examples of ARKode Usage

ARKode has been freely-available since 2014. We have specifically worked with applications groups in:

ParaDiS – large-scale simulations of dislocation growth/propagation (material strain hardening)

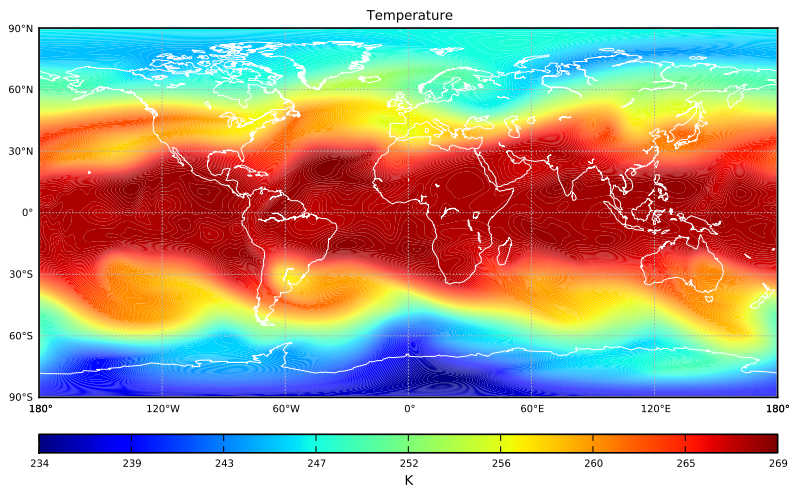
- Examined high-order adaptive DIRK methods.
- Examined nonlinear solvers and options.
- *51%-98% speedup (production/test problems)*



Gardner et al., *MSMSE*, 2015

Tempest & **HOMME-NH** – non-hydrostatic 3D dynamical cores for atmospheric simulations

- Examined IMEX splittings & fixed-step ARK methods for accuracy/stability
- Examined nonlinear/linear solver algorithms for implicit components
- *Identified 'optimal' splitting+solver+ARK; invented new climate-specific ARK methods.*



Gardner et al., *GMD*, 2018; Vogl et al, 2019.

Reconfiguring ARKode into an infrastructure

Over the last 18 months, we have overhauled ARKode to serve as an infrastructure for general, adaptive, one-step time integration methods:

- ARKode provides the outer time integration loop and generic usage modes (interpolation vs "tstop"; one-step versus time interval).
- Time-stepping modules handle problem-specific components: definition of the IVP, algorithm for a single time step.
- Time-stepping modules may leverage shared ARKode infrastructure:
  - SUNDIALS' vector, matrix, linear solver and nonlinear solver objects,
  - translation between generic solvers and IVP-specific algebraic systems,
  - time-step adaptivity controllers: PID, PI, I, *user-supplied*,
  - ...

Increased agility for implementing state-of-the-art algorithms in a production software environment.

ARKode Time-Stepping Modules

The new ARKode structure based on time-stepping modules eases the implementation of new integration schemes:

- **ARKStep** Provides all the functionality from previous ARKode versions. Supports ARK, DIRK and ERK methods for problems of the form

$$M\dot{y} = f^E(t,y) + f^I(t,y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0.$$

- **ERKStep**: A streamlined module that provides more optimal support for ERK-specific methods applied to the standard IVP form

$$\dot{y} = f(t,y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0.$$

- **MRIStep**: Provides support for Multirate Infinitesimal Step (MIS) like methods

$$\dot{y} = f^S(t,y) + f^F(t,y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0.$$

- **IMEXGARKStep**: Provides support for IMEX Generalized Runge-Kutta methods (currently testing)

$$M\dot{y} = f^E(t,y) + f^I(t,y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0.$$

Multirate Infinitesimal Step (MIS) methods [Knoth & Wolke 1998; Schlegel et al. 2009; ...]

MIS methods arose in the numerical weather prediction community. This generic infrastructure supports $\mathcal{O}(h^2)$ and $\mathcal{O}(h^3)$ methods for multirate problems:

$$\dot{y} = f^S(t,y) + f^F(t,y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0.$$

- $f^F(t,y)$ contains the "fast" terms; $f^S(t,y)$ contains the "slow" terms.
- $h^S > h^F$, with a time scale separation $h^S/h^F \approx m$.
- y is frequently partitioned as well, e.g. $y = \left[y^F\ y^S\right]^T$.
- The slow component may be integrated using an explicit "outer" RK method, $T_O = \{A, b, c\}$, where $c_i \leq c_{i+1}$, $i = 1, \ldots, s-1$.
- The fast component is advanced between slow stages as the exact solution of a modified ODE.
- Practically, this fast solution is subcycled using an "inner" RK method.

MIS Algorithm

Denoting $y_n \approx y(t_n)$, a single MIS step $y_n \to y_{n+1}$ is as follows:

1. Set $z_1 = y_n$
2. For each outer Runge-Kutta stage $z_i, i = 2, \ldots, s+1$:

   a) Let $v(t_{n,i-1}) = z_{i-1}$ and $r = \sum_{j=1}^{i-1} \left(\frac{A_{i,j} - A_{i-1,j}}{c_i - c_{i-1}} \right) f^S\left(t_{n,j}, z_j\right)$

   b) Solve the fast ODE: $\dot{v}(\tau) = f^F\left(\tau, v\right) + r$, for $\tau \in [t_{n,i-1}, t_{n,i}]$

   c) Set $z_i = v(t_{n,i})$
3. Set $y_{n+1} = z_{s+1}$

where the outer stage times are $t_{n,j} = t_n + c_j h^S$ and $A_{s+1,j} = b_j$.

- When $c_i = c_{i+1}$, the IVP "solve" reduces to a standard Runge–Kutta update.
- Step 2b may use any applicable algorithm of sufficient accuracy.

MIS Properties

MIS methods satisfy a number of desirable multirate method properties:

- The MIS method is $\mathcal{O}\left(h^2\right)$ if both inner/outer methods are at least $\mathcal{O}\left(h^2\right)$.
- The MIS method is $\mathcal{O}\left(h^3\right)$ if both inner/outer methods are at least $\mathcal{O}\left(h^3\right)$, and T_O satisfies

$$\sum_{i=2}^{s} \left(c_i - c_{i-1}\right)\left(e_i + e_{i-1}\right)^T Ac + \left(1 - c_s\right)\left(\frac{1}{2} + e_s^T Ac\right) = \frac{1}{3}.$$

- The inner method may be a subcycled T_O, enabling a *telescopic* multirate method (i.e., n-rate problems supported via recursion).
- Both inner/outer methods can utilize problem-specific tables (SSP, etc.).
- h^F may be varied within a slow step to adapt the multirate structure.
- Highly efficient – only a single traversal of $[t_n, t_{n+1}]$ is required. To our knowledge, MIS are the most efficient $\mathcal{O}\left(h^3\right)$ multirate methods available.

MRIStep Module

The new MRIStep module supports $\mathcal{O}(h^2)$ and $\mathcal{O}(h^3)$ MIS-like methods.

- The slow time scale is integrated with an explicit Runge–Kutta method.
- The slow time scale uses a user-defined h^S that can be varied between slow steps.
- The fast time scale is advanced by calling the ARKStep module, and thus allows for explicit, implicit or IMEX integration.
- The fast time scale can use adaptive or fixed time step sizes.
- Supports user-defined Butcher tables for both time scales.
- Explicit-explicit released in SUNDIALS v4.0, implicit and IMEX fast are available in SUNDIALS v5.0 beta releases.

Currently implementing extensions to $\mathcal{O}(h^4)$ and implicit slow integration.

MRIStep Example – Advection-Reaction

Explicit (slow) advection and implicit (fast) Brusselator reactions:

$$u_t = -cu_x + k_1 A - k_2 Bu + k_3 u^2 v - k_4 u$$
$$v_t = -cv_x + k_2 Bu - k_3 u^2 v$$

Comparison of $\mathcal{O}(h^3)$ MIS vs IMEX (time steps chosen for comparable accuracy):

	IMEX	**MRI**
Time steps	6,805	1,429 / 8,813
Advection evals	29,379	4,288
Reaction evals	98,917	116,949

- MIS requires fewer advection evaluations, lowering MPI communication costs.
- MRIStep benefits depend on an array of factors:
  - separation of time scales (e.g., CFL vs reaction rates)
  - relative cost of f^S vs f^F function evaluations

MRIStep Example in AMReX (from J. Loffeld @ LLNL)

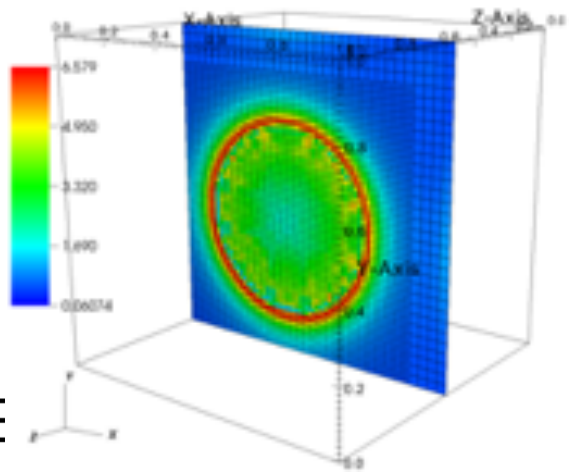
We have integrated SUNDIALS with AMReX, providing a powerful new set of tools for solving multiscale problems posed on block-structured AMR grids.

- `NVector` API enables SUNDIALS to use native AMReX structures for single-level and multi-level problems.
- `SUNLinearSolver` API enables SUNDIALS to use AMReX `MLABecLaplacian` solver for scalable diffusion solves in implicit methods.

3D Brusselator (two-level AMR grid with 2:1 refinement ratio):

- Explicit-explicit MRIStep with $h^S/h^F = 100$.
- MRIStep effectively removes advection cost for similar accuracy requirements.
- At core-counts tested, advection costs 66% more than reactions (will increase at larger scales)

2D slice of 3D Brusselator solution



Results from LLNL's Quartz cluster:

Cores	Fine Grid	Time (s)
64	256^3	335
512	512^3	342
4096	1024^3	368

Extensions to Higher Order and Implicit Slow

We are currently exploring extensions for $\mathcal{O}\left(h^4\right)$ methods in MRIStep:

- MRI-GARK methods [Sandu, 2018] modify the MIS slow forcing term in the fast IVP to have time-dependent coefficients:

$$r(\tau) = \sum_{j=1}^{i-1} \gamma_{i,j}\left(\frac{\tau - t_{n,i-1}}{(c_i - c_{i-1})h^S}\right) f^S\left(t_{n,j}, z_j\right).$$

- MRI-GARK also include *decoupled implicit methods* that do not require solving a fully coupled fast and slow system – these can readily leverage the existing ARKStep solver infrastructure.

- Relaxed MIS methods (RMIS) [Sexton & R., 2018] compute y_{n+1} as a combination of $\{f^F(t_{n,i}, z_i) + f^S(t_{n,i}, z_i)\}$ to produce a $\mathcal{O}\left(h^4\right)$ method with $\mathcal{O}\left(h^3\right)$ MIS embedding.

- Multirate exponential Runge–Kutta methods (MERK) [Luan, Chinomona & R., 2019] are similar to MRI-GARK, constructing time-dependent fast IVP forcing terms $r(\tau)$, but rely on exponential integrator theory (as opposed to GARK).

Conclusions

The ARKode infrastructure flexibly supports extensive studies of optimal algorithms for multiphysics problems:

- Numerous built-in methods; supports user-supplied.
- Numerous vector/matrix data structures, support for user-supplied.
- Numerous algebraic solver algorithms, support for user-supplied.
- Actively developing state-of-the-art flexible time integration methods for multiphysics applications:
  - Additive partitioning – break apart physical processes based on stiffness (implicit/explicit/IMEX) or time scale (fast/slow).
  - New support for explicit slow MIS methods with explicit, implicit or IMEX fast integration, with higher order and implicit slow coming.
  - Focus on ease-of-use and support for user-supplied components, allowing SUNDIALS to reuse previous development of optimal data structures and algebraic solvers.

Thanks & Acknowledgements

- Carol S. Woodward [LLNL]
- Vu Thai Luan [Mississippi State]
- John Loffeld [LLNL]
- Cody J. Balos [LLNL]
- Alan C. Hindmarsh [LLNL]
- DOE SciDAC & ECP Programs
- SMU Center for Scientific Computation

Software:

- ARKode/SUNDIALS – https://computation.llnl.gov/casc/sundials
- AMReX – https://amrex-codes.github.io

Support:

- U.S. Department of Energy Office of Science project *Frameworks, Algorithms and Scalable Technologies for Mathematics (FASTMath)*, under Lawrence Livermore National Laboratory subcontract B626484.
- *Exascale Computing Project* (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

References

- Ropp & Shadid, *J. Comput. Phys.*, 203, 2005.
- Estep et al., *Comput. Meth. Appl. Mech. Eng.*, 196, 2007.
- Ascher et al., *Applied Numerical Mathematics*, 25, 1997.
- Araújo et al., *SIAM J. Numer. Anal.*, 34, 1997.
- Gardner et al., *Model. Simul. Mater. Sci. Eng.*, 23, 2015.
- Gardner et al., *Geosci. Model Dev.*, 11, 2018.
- Vogl et al., *arXiv:1904.10115*, 2019.
- Knoth & Wolke, *Appl. Numer. Math.*, 1998.
- Schlegel et al., *J. Comput. Appl. Math.*, 2009.
- Sandu & Günther, *SINUM.*, 2015.
- Sexton & Reynolds, *arXiv:1808.03718*, 2018.
- Sandu, *arXiv:1808.02759*, 2018.
- Luan, Chinomona & Reynolds, *arXiv:1904.06474*, 2019.

Disclaimer