Flexible, accurate and scalable time integration of multiphysics problems

Daniel R. Reynolds

reynolds@smu.edu

Department of Mathematics, Southern Methodist University

Computational Mathematics, Science and Engineering Colloquium
Michigan State University
14 September 2020

Collaborators & Funding

Collaborators:

- **LLNL SUNDIALS team: D. Gardner, C. Woodward, C. Balos, A. Hindmarsh**
- SMU students: R. Chinomona, T. Yan
- Missippi State: V. Luan
- AMReX/PELE team: J. Sexton, A. Felden (LBL); H. Sitaraman (NREL)
- Climate: C. Vogl (LLNL); M. Taylor, A. Steyer (SNL); P. Ullrich, J. Guerra (UC Davis); J. Pudykiewicz (Env. Canada)
- MGK team: D. Ernst (MIT), M. Francisquez (PPPL)
- Enzo team: M. Norman, J. Bordner (UCSD); B. O'Shea (MSU), . . .

Grant/Computing Support:

- DOE SciDAC, ECP & INCITE Programs
- NSF AST & XSEDE Programs
- SMU Center for Scientific Computation
- DOD DURIP Program

Multiphysics Scientific Simulations

In recent decades computation has rapidly assumed its role as the third pillar of the scientific method [Vardi, *Commun. ACM*, 53(9):5, 2010]:

- Simulation complexity has evolved from simplistic calculations of only 1 or 2 basic equations, to massive models that combine vast arrays of processes.
- Early algorithms could be analyzed using standard techniques, but mathematics has not kept up with the fast pace of scientific simulation development.
- Presently, many numerical analysts construct elegant solvers for models of limited practical use, while computational scientists "solve" highly-realistic systems using *ad hoc* methods with questionable reliability.

We are working to bridge this gap between mathematical theory and computing practice.

Outline

1. Motivation
2. "Flexible" Integrators
3. Applications
4. Conclusions

Outline

1 Motivation
- Climate
- Cosmology
- Fusion
- Legacy Methods

2 "Flexible" Integrators

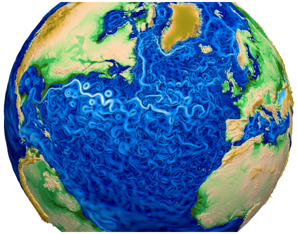
3 Applications

4 Conclusions

Climate – Energy Exascale Earth System Model (E3SM)

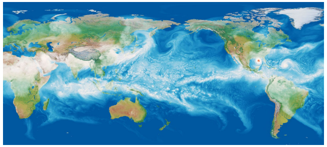
Motivation: 2013 DOE report on need for climate model predictions of energy sector impacts:

- air and water temperature trends
- water availability
- storms and heavy precipitation
- coastal flooding and sea-level rise



Mission (`https://e3sm.org/about/vision-and-mission`)

- integrate advanced models and algorithms to push the high-resolution frontier
- bridge the gap in modeling scales and processes to include natural, managed and man-made systems
- develop ensemble modeling strategies to quantify uncertainty



`https://e3sm.org`

[In collab. w/ D. Gardner, C. Vogl & C. Woodward (LLNL); A. Steyer & M. Taylor (SNL), P. Ullrich & J. Guerra (UC Davis)]

Nonhydrostatic Atmospheric Models

- Increased computational power enables spatial resolutions beyond the hydrostatic limit.
- Nonhydrostatic models consider the 3D compressible Navier Stokes equations; these support acoustic (sound) waves.
- Acoustic waves have a negligible effect on climate, but travel much faster than convection (343 m/s vs 100 m/s horizontal and 1 m/s vertical), leading to overly-restrictive explicit stability restrictions.

- To overcome this stiffness, nonhydrostatic models utilize split-explicit, implicit-explicit, or fully implicit time integration.
- Additionally, climate "dycores" are coupled to myriad other processes (ocean, land/sea ice, . . .), each evolving on significantly different time scales.

Nonhydrostatic Formulation (Tempest)

[Gardner, Guerra, Hamon, R., Ullrich & Woodward, 2018]

Tempest is an experimental dycore used for method development; it considers 5 governing [hyperbolic] equations in an arbitrary coordinate system:

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{1}{J}\frac{\partial}{\partial \alpha}\left(J\rho u^{\alpha}\right) - \frac{1}{J}\frac{\partial}{\partial \beta}\left(J\rho u^{\beta}\right) - \frac{1}{J}\frac{\partial}{\partial \xi}\left(J\rho u^{\xi}\right)\\
\frac{\partial u_\alpha}{\partial t} &= -\frac{\partial}{\partial \alpha}(K+\Phi) - \theta\frac{\partial \Pi}{\partial \alpha} + (\eta \times \mathbf{u})_\alpha\\
\frac{\partial u_\beta}{\partial t} &= -\frac{\partial}{\partial \beta}(K+\Phi) - \theta\frac{\partial \Pi}{\partial \beta} + (\eta \times \mathbf{u})_\beta\\
\left(\frac{\partial r}{\partial \xi}\right)\frac{\partial w}{\partial t} &= -\frac{\partial}{\partial \xi}(K+\Phi) - \theta\frac{\partial \Pi}{\partial \xi} + u^{\alpha}\frac{\partial u_\alpha}{\partial \xi} + u^{\beta}\frac{\partial u_\beta}{\partial \xi} - u^{\alpha}\frac{\partial u_\xi}{\partial \alpha} - u^{\beta}\frac{\partial u_\xi}{\partial \beta}\\
\frac{\partial \theta}{\partial t} &= -u^{\alpha}\frac{\partial \theta}{\partial \alpha} - u^{\beta}\frac{\partial \theta}{\partial \beta} - u^{\xi}\frac{\partial \theta}{\partial \xi},
\end{aligned}
$$

where ρ is the density, (u_α, u_β) are the horizontal velocity, w is the vertical velocity, and θ is the potential temperature. *Key: horizontal propagation and vertical propagation.*

SMU ECP EXASCALE COMPUTING PROJECT FASTMATH sundials

Nonhydrostatic Formulation (HOMME-NH) [Vogl, Steyer, R., Ullrich & Woodward, 2019]

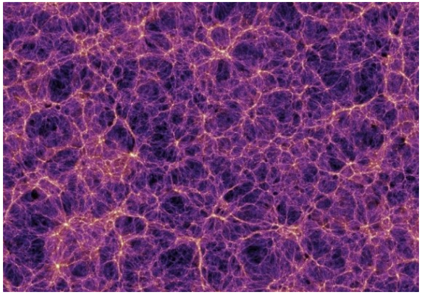
HOMME-NH will be the "production" dycore in E3SM v2 responsible for global atmospheric flow:

$$
\begin{aligned}
\frac{\partial}{\partial t}\left(\frac{\partial \pi}{\partial \eta}\right) &= -\nabla_\eta \cdot \left(\frac{\partial \pi}{\partial \eta}\mathbf{u}\right) - \frac{\partial}{\partial \eta}\left(\pi \frac{\mathrm{d}\eta}{\mathrm{d}t}\right) \\
\frac{\partial \mathbf{u}}{\partial t} &= -(\nabla_\eta \times \mathbf{u} + 2\mathbf{\Omega}) \times \mathbf{u} - \frac{1}{2}\nabla_\eta (\mathbf{u}\cdot\mathbf{u}) - \frac{\mathrm{d}\eta}{\mathrm{d}t}\frac{\partial \mathbf{u}}{\partial \eta} - \frac{1}{\rho}\nabla_\eta p \\
\frac{\partial w}{\partial t} &= -\mathbf{u}\cdot\nabla_\eta w - \frac{\mathrm{d}\eta}{\mathrm{d}t}\frac{\partial w}{\partial \eta} - g(1-\mu), \quad \mu = \left(\frac{\partial p}{\partial \eta}\right) / \left(\frac{\partial \pi}{\partial \eta}\right), \\
\frac{\partial \Theta}{\partial t} &= -\nabla_\eta \cdot (\Theta \mathbf{u}) - \frac{\partial}{\partial \eta}\left(\Theta \frac{\mathrm{d}\eta}{\mathrm{d}t}\right), \quad \Theta = \frac{\partial \pi}{\partial \eta}\theta, \\
\frac{\partial \phi}{\partial t} &= -\mathbf{u}\cdot\nabla_\eta \phi - \frac{\mathrm{d}\eta}{\mathrm{d}t}\frac{\partial \phi}{\partial \eta} + gw,
\end{aligned}
$$

where π is hydrostatic pressure, η is vertical coordinate, $\mathbf{u}$ and w are horizontal and vertical velocities, θ is potential temperature, and ϕ is geopotential. *Key: hydrostatic model and nonhydrostatic terms.*

Cosmic Reionization – The Origins of the Universe

- After the Big Bang, primordial matter (96% dark matter, 2.92% H, 1% He) was strewn throughout the universe.
- Gravitational attraction condensed this into the "cosmic web," the large-scale structure that connects/creates galaxies.



[http://svs.gsfc.nasa.gov/cgi-bin/details.cgi?aid=10118]

- Each bright spot above is an entire galaxy; purple filaments show where material connects these. To the eye, only the galaxies are visible.
- This visualization spans 134 Mpc (437 million light-years) per side.

[In collab. w/ M. Norman & J. Bordner (UCSD), B. O'Shea (MSU), J. Wise (GA Tech) and the rest of the *ENZO* team]

Cosmology Multiphysics Model

[Bryan et al., 1995; R. et al., 2009; Norman, R. & So, 2009; Bryan et al., 2014]

Cold dark matter motion ($k = 1, \ldots, N_d$), cosmological expansion:

$$\mathbf{q}'_{d,k}(t) = \mathbf{v}_{d,k}, \qquad \mathbf{v}'_{d,k}(t) = -\frac{1}{m_{d,k}}\nabla\phi,$$

$$\nabla^2\phi = \frac{4\pi G}{a}\left[\rho_b + \rho_d(\mathbf{q_d}) - \rho_0\right],$$

$$\frac{a''(t)}{a} = -\frac{4\pi G}{3a^3}\left(\rho_0 + 3\frac{p_0}{c^2}\right) + \frac{\Lambda}{3}, \qquad \mathbf{x} \equiv \frac{\mathbf{r}}{a(t)}.$$

Hydrodynamic motion (conservation of mass, momentum and energy):

$$\partial_t\rho_b + \frac{1}{a}\mathbf{v}_b\cdot\nabla\rho_b = -\frac{1}{a}\rho_b\nabla\cdot\mathbf{v}_b,$$

$$\partial_t\mathbf{v}_b + \frac{1}{a}\left(\mathbf{v}_b\cdot\nabla\right)\mathbf{v}_b = -\frac{a'}{a}\mathbf{v}_b - \frac{1}{a\rho_b}\nabla p - \frac{1}{a}\nabla\phi,$$

$$\partial_t e + \frac{1}{a}\mathbf{v}_b\cdot\nabla e = -\frac{2a'}{a}e - \frac{1}{a\rho_b}\nabla\cdot\left(p\mathbf{v}_b\right) - \frac{1}{a}\mathbf{v}_b\cdot\nabla\phi.$$

Multi-frequency radiation transport & chemical ionization:

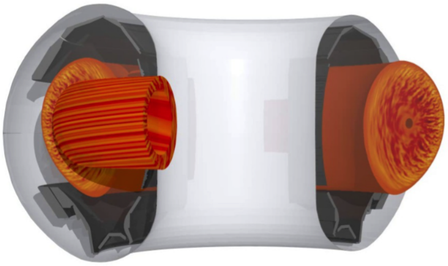
$$\partial_t E_\nu + \nabla\cdot\left(E_\nu\mathbf{v}_b\right) - \nabla\cdot\left(D\,\nabla E_\nu\right) = \frac{\nu a'}{a}\partial_\nu E_\nu - \frac{3a'}{a}E_\nu + \eta_\nu - c\kappa_\nu E_\nu, \quad \nu = 1,\ldots,N_f,$$

$$\partial_t\mathbf{n}_i + \nabla\cdot\left(\mathbf{n}_i\mathbf{v}_b\right) = -\mathbf{n}_i\Gamma_i^{ph} + \alpha_{i,j}^{rec}\mathbf{n}_e\mathbf{n}_j, \quad i,j = 1,\ldots,N_c.$$

Fusion Plasma Simulations

Large-scale, nonlinear simulation of fusion plasmas is critical for the design of next-generation confinement devices.

- Fusion easy to achieve but difficult to *stabilize*, as needed to increase yield and protect device.
- Linear modes present in fluid models are typically well-controlled.
- Most current work focuses on disruptions due to nonlinear instabilities and kinetic effects.
- Turbulence in the sharp edge disrupts the core, but is difficult to simulate:
  - must accurately couple ions and electrons in high dimensions: $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{v} \in \mathbb{R}^d$, $t \in \mathbb{R}$; $d = \{2, 3\}$
  - mass/velocity differences result in $100\times$ spatial/temporal scale separation.


GENE gyrokinetic simulation of core turbulence

[In collab. w/ D. Ernst (MIT); M. Francisquez (PPPL) and the rest of the MGK SciDAC Project]

Multiscale Model for Ion/Electron Turbulence Interactions

Before tackling full 5D gyrokinetic turbulence with GENE, we are investigating high-order multirate methods for a reduced pseudospectral model for ITG/ETG turbulence:

$$\begin{aligned}
\frac{\partial n_e}{\partial t} + [\Psi, n_e] - i\omega_{*e}\Psi + 2i\omega_{de}\Psi + i\omega_{de}\left(2n_e + T_{\perp e1}\right) = 0,\\
\frac{\partial T_{\perp e}}{\partial t} - \left(1+\eta_{\perp e}\right) i\omega_{*e}\Psi + \left[\frac{1}{2}\hat{\nabla}_{\perp}^2\Psi, n_e\right] + [\Psi, T_{\perp e}] + 3i\omega_{de}\Psi = 0,\\
\frac{\partial n_i}{\partial t} + [\Psi, n_i] - \frac{1}{\tau} i\omega_{*i}\Psi + \frac{2}{\tau} i\omega_{di}\Psi + i\omega_{di}\left(2n_i + \tau^{-1}T_{\perp i}\right) = 0,\\
\frac{\partial T_{\perp i}}{\partial t} - \left(1+\eta_{\perp i}\right) i\omega_{*i}\Psi + \tau\left[\frac{1}{2}\hat{\nabla}_{\perp}^2\Psi, n_i\right] + [\Psi, T_{\perp i}] + 3i\omega_{di}\Psi = 0.
\end{aligned}$$

We evolve equations in frequency space, but convert to/from real space for computing the nonlinear Poisson brackets.

A Sampling of Multiphysics Challenges

These multiphysics problems exhibit key characteristics that challenge traditional numerical methods:

- "Multirate" structure: different processes evolve on distinct time scales, but these are too close to analytically reformulate (e.g., via steady-state approximation).
- The existence of stiff components prohibits fully explicit methods.
- Nonlinearity and insufficient differentiability challenge fully implicit methods.
- "Multiscale" structure: some spatial regions may be well-modeled via coarse meshes, while others require high resolution.
- Extreme parallel scalability demands optimal algorithms. While robust and scalable algebraic solvers exist for some pieces (e.g., FMM for particles, multigrid for diffusion), none optimal for the full combination.

We have obviously not solved all of the above problems, I only point them out to highlight the work ahead.

"Classical" Time Integrators (and their deficiencies)

Historically, IVP research has focused on two simple problem types:

$$\begin{aligned} y'(t) &= f(t, y(t)), && y(t_0) = y_0 && \text{[ODE]} \\ 0 &= F(t, y(t), y'(t)), && y(t_0) = y_0, \quad y'(t_0) = y_0' && \text{[DAE]} \end{aligned}$$

Corresponding solvers thus enforced overly-rigid standards:

- Treat all components implicitly or explicitly, without IMEX flexibility.
  - Fully explicit: "stiff" components require overly-small time steps for stability.
  - Fully implicit: scalable/robust algebraic solvers difficult for highly nonlinear or nonsmooth terms.
- Inflexible vector/matrix/solver data structures. While contiguous 1D vectors and matrices work well in LAPACK/MATLAB, these are rarely optimal for large-scale, multiphysics problems.
- Software was hard-coded for specific methods and parameters – while these are decent for most problems, they're rarely optimal for any.

Ad Hoc Algorithms Pervade Scientific Computing Applications

On the other hand, practitioners frequently "split" their problems and solve each component separately over a time step $[t_0, t_0 + h]$, e.g. a "Lie-Trotter" splitting:

$$\begin{aligned} & y'(t) = f_1(t,y) + \cdots + f_m(t,y), \quad y(t_0) = y_0 \\ \approx \quad & \\ & y_1'(t) = f_1\left(t, y_1\right), \qquad y_1(t_0) = y_0, \\ & \qquad \vdots \\ & y_m'(t) = f_m\left(t, y_m\right), \quad y_m(t_0) = y_{m-1}(t_0 + h), \end{aligned}$$

While each component may be tackled independently (or even subcycled) using, e.g., something from "Numerical Recipes," the overall approach suffers from:

- Low accuracy – typically only $\mathcal{O}(h)$; 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].

Filling this 'Disconnect' between Mathematical Software and Multiphysics Practice

We work to construct flexible time integration methods, disseminated as robust open-source software, to improve temporal integration of multiphysics systems.

Goals:

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

Outline

1 Motivation

2 "Flexible" Integrators
- ARK IMEX methods
- [Practical] exponential integrators
- Multirate methods
- Architectural flexibility

3 Applications

4 Conclusions

Additively-split Multiphysics Models

Modern time-integration methods focus on high-order accuracy and increased numerical stability for multiphysics systems in additively-partitioned form:

$$y'(t) = f_1(t,y) + \cdots + f_m(t,y), \quad y(t_0) = y_0.$$

Note that 'variable partitioned' problems,

$$\begin{aligned} y_1'(t) &= \hat{f}_1\left(t,y\right), \qquad y_1(t_0) = y_{1,0},\\ &\vdots\\ y_m'(t) &= \hat{f}_m\left(t,y\right), \quad y_m(t_0) = y_{m,0}, \end{aligned}$$

are automatically included through appropriate partitioning of $y = \begin{bmatrix} y_1 & \cdots & y_m \end{bmatrix}^T$ and $f_i(t,y) = \begin{bmatrix} 0 & \cdots & 0 & \hat{f}_i(t,y) & 0 & \cdots & 0 \end{bmatrix}^T$.

Additive Runge–Kutta (ARK) Methods [Ascher et al. 1997; Araújo et al. 1997; Kennedy & Carpenter 2003; ...]

In 2014, we released the ARKODE package as part of SUNDIALS, providing adaptive ARK methods for mixed implicit-explicit calculations:

$$M(t)\,y'(t) = f^E(t,y) + f^I(t,y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0,$$

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

Combine two s-stage RK methods; denoting $h_n = t_{n+1} - t_n$, $t^E_{n,j} = t_n + c^E_j h_n$, $t^I_{n,j} = t_n + c^I_j h_n$:

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

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

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

Solving each stage z_i, $i = 1, \ldots, s$

Per-stage cost is commensurate with implicit Euler for $y'(t) = f^I(t, y)$ – solve a root-finding problem:

$$0 = G_i(z) = \left[z - h_n a^I_{i,i} f^I(t^I_{n,i}, z)\right] - \left[y_n + h_n \sum_{j=1}^{i-1} \left(a^E_{i,j} f^E(t^E_{n,j}, z_j) + a^I_{i,j} f^I(t^I_{n,j}, z_j)\right)\right]$$

- If $f^I(t, y)$ is *linear* in y then this is a large-scale linear system for each z_i.
- Else G_i is nonlinear, requiring an iterative solver – ARKODE supports Newton, accelerated fixed-point, or customized (problem-specific) methods.

In recent years, we have enhanced ARKODE in a number of ways to now include a variety of ‘steppers’:

- ARKSTEP: this supports all functionality originally included in ARKODE (ARK methods).
- ERKSTEP: tuned for highly efficient explicit Runge–Kutta methods.
- MRISTEP: new ‘multirate’ time stepping module (more on this in a few slides).

Exponential Rosenbrock (ExpRB) Methods [Hochbruch et al., 2009; Luan & Ostermann, 2014]

Exponential Rosenbrock methods consider a specific additive splitting of the IVP:

$$y'(t) = f(y) = \mathcal{J}(y)y + \mathcal{N}(y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0,$$

- $\mathcal{J}(y) \equiv \frac{\partial f(y)}{\partial y}$ is the Jacobian of the full right-hand side, f [assumed stiff], and
- $\mathcal{N}(y) \equiv f(y) - \mathcal{J}(y)y$ contains any remaining nonlinearities [assumed nonstiff].

Analytical solution over $t \in [t_n, t_n + h]$ uses the variation-of-constants formula:

$$y(t) = \mathrm{e}^{(t-t_n)\mathcal{J}(y_n)}y(t_n) + \int_0^t \mathrm{e}^{(t-\tau)\mathcal{J}(y_n)}\mathcal{N}(u(t_n+\tau))\mathrm{d}\tau.$$

By approximating the integral via quadrature, an s-stage ExpRB method may be written:

$$z_i = y_n + c_i h\,\varphi_1(c_i h\mathcal{J}_n)f(y_n) + h\sum_{j=2}^{i-1} a_{ij}(h\mathcal{J}_n)(\mathcal{N}_n(z_j) - \mathcal{N}_n(y_n)),$$

$$y_{n+1} = y_n + h\,\varphi_1(h\mathcal{J}_n)f(y_n) + h\sum_{i=2}^{s} b_i(h\mathcal{J}_n)(\mathcal{N}_n(z_i) - \mathcal{N}_n(y_n))$$

where $z_1 = y_n$, $\mathcal{J}_n \equiv \mathcal{J}(y_n)$, $\mathcal{N}_n \equiv \mathcal{N}(y_n)$, and $\varphi_1(z) \equiv (e^z - 1)/z$.

ExpRB Implementation [Luan & Ostermann, 2014; Niesen & Wright, 2012; Luan, Pudykiewicz & R., 2019]

$a_{ij}(h\mathcal{J}_n), b_i(h\mathcal{J}_n)$ are linear comb. of the matrix functions $\varphi_k(c_i h\mathcal{J}_n), \varphi_k(h\mathcal{J}_n)$, resp.; defined recursively via

$$\varphi_{k+1}(z) \equiv \frac{\varphi_k(z) - 1/k!}{z}, \quad k \geq 1.$$

The primary challenge in applying ExpRB methods is efficiently computing linear combinations of these matrix functions multiplied by vectors,

$$w_k = \sum_{l=0}^{p} \varphi_l(c_k A)\, v_l, \quad k = 2, \ldots, s,$$

where each $c_k \in (0, 1]$ denotes a "time" scaling factor used for the output w_k.

- Modern approaches exploit the structure of these φ_k functions to construct efficient implementations that require no matrix factorizations.
- In 2019, we released a prototype MATLAB implementation tuned for ExpRB methods as the `phipm_simul_iom` algorithm. We hope to extend this to a new ARKODE module in the near future.

Multirate Infinitesimal Step (MIS/MRI) methods [Schlegel et al. 2009; Sandu 2019; . . .]

MRI methods arose in the numerical weather prediction community. This generic infrastructure supports up to $\mathcal{O}(h^4)$ methods for multirate problems:

$$y'(t) = f^S(t,y) + f^F(t,y), \quad t \in [t_0, t_f], \quad y(t_0) = y_0.$$

- $f^S(t,y)$ contains the “slow” dynamics, integrated with time step H.
- $f^F(t,y)$ contains the “fast” dynamics, integrated with time step $h \ll H$
- The slow component is integrated using an “outer” RK method, while the fast component is advanced between slow stages by solving a modified ODE with a subcycled “inner” RK method.
- Highly efficient – requires only a single traversal of $[t_n, t_{n+1}]$ for high order methods.

MRI Algorithm

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

1. Set $z_1 = y_n$
2. For each slow Runge-Kutta stage $z_i, i = 2, \ldots, s+1$:
   a) Let $v(t_{n,i-1}) = z_{i-1}$ and $r(\tau) = \frac{1}{\Delta c_i} \sum_{j=1}^{i} \gamma_{i,j}\left(\frac{\tau - t_{n,i-1}}{\Delta c_i H}\right) f^S\left(t_{n,j}, z_j\right)$
   b) Solve the fast ODE: $v'(\tau) = f^F\left(\tau, v\right) + r(\tau)$, 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 $t_{n,j} = t_n + c_j H$ and $\Delta c_i = c_i - c_{i-1}$.

- $\gamma_{i,j}(\theta)$ is a polynomial in θ, with coefficients that derive from the slow Runge–Kutta method.
- When $c_i = c_{i+1}$, the IVP "solve" reduces to a standard ERK/DIRK Runge–Kutta update.
- Step 2b may use any applicable algorithm of sufficient accuracy.

MRISTEP Module in ARKODE

The current MRISTEP module in ARKODE (v4.3.0) supports:

- $\mathcal{O}(H^2)$ and $\mathcal{O}(H^3)$ order explicit-slow MIS methods with fixed slow step sizes.
- Fast time scale is evolved with ARKSTEP (explicit, implicit or IMEX), with adaptive or fixed step sizes.

Upcoming ARKODE release will support solve-decoupled implicit methods: alternate between subcycling steps $(\gamma_{i,i} = 0, \Delta c_i \neq 0)$ and standard DIRK steps $(\gamma_{i,i} \neq 0, \Delta c_i = 0)$.

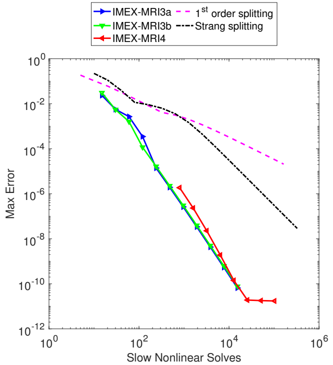
Currently implementing up to $\mathcal{O}(H^4)$ IMEX-MRI methods that support IMEX treatment of slow time scale [Chinomona and R., 2020]:

$$y'(t) = f^E(t,y) + f^I(t,y) + f^F(t,y), \quad y(t_0) = y_0.$$

Right: 1D advection-diffusion-reaction example – IMEX-MRI shows significant efficiency improvements over Lie-Trotter and Strang-Marchuk.

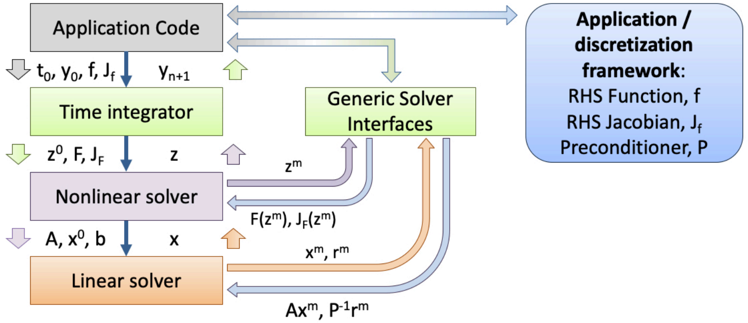
Also deriving higher-order multirate approaches:

- Multirate Exponential Runge–Kutta (MERK) allow $\mathcal{O}(H^5)$ [Luan, Chinomona and R., 2020]
- Multirate Exponential Rosenbrock (MERB) allow $\mathcal{O}(H^6)$ [Luan, Chinomona and R., *in prep*]



SUNDIALS' Modular Design & Control Inversion

Control passes between integrator, solvers, and application code as the integration progresses:



- Control passes from the integrator to the solvers and application code as the integration progresses
- Time integrator and nonlinear solver are agnostic of vector data layout and specific solvers used

Multiphysics Enhancements

SUNDIALS includes numerous additional enhancements for multiphysics codes:

- Packages modify solution data only through the *NVector* class:
- Several optional implementations are released with SUNDIALS:
    - CUDA, RAJA, and OpenMPDEV (target offload) vectors provide GPU support *(HIP, RAJA+HIP and Kokkos are coming soon)*.
    - Parallel, ParHyp (*hypre*), PETSc, and Trilinos modules are MPI distributed.
    - ManyVector and MPIPlusX modules provide support for hybrid computation.
- Application-specific vectors, matrices, linear and even nonlinear solvers may be easily supplied.
- Current release includes fully-featured Fortran 2003 interfaces for all packages.
- ARKODE-specific multiphysics enhancements:
    - Many built-in RK tables, adaptivity controllers & implicit predictors; supports user-supplied modules.
    - Ability to resize data structures based on changing IVP size (AMR).
    - All internal solver parameters are user-modifiable.

Outline

1 Motivation

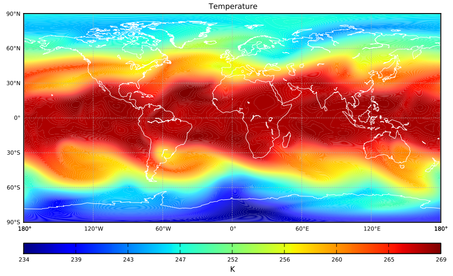
2 "Flexible" Integrators

3 Applications
- Climate
- Multiphysics/Multirate Testing
- GPU Chemistry

4 Conclusions

ARK IMEX methods in climate [Gardner et al., 2018; Vogl et al., 2019; Ullrich et al., 2018]

We explored optimal ARK IMEX methods for next-generation nonhydrostatic climate codes *Tempest* & *HOMME-NH*.



Examined:

- 5 IMEX splittings; 21 published & 13 custom ARK methods (optimized explicit stability along imaginary axis).
- Various algebraic solvers for implicit components.
- Effects of "standard" stabilization approaches (hyperviscosity, vertical remap).

Findings:

- ARKODE's modular structure allowed rapid exploration of "solver space."
- Stability $\propto$ implicitness, but horizontally implicit terms *significantly* increase cost.
- Best overall ARK methods were those we designed for this application.
- Linearly-implicit solves work for current h, but nonlinearity increasingly relevant at desired h.

ExpRB methods with the 2D shallow water equations [Luan, Pudykiewicz & R., 2019]

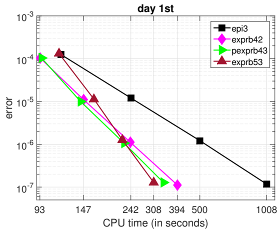
Simplified 2D SWE model used in climate and weather prediction:

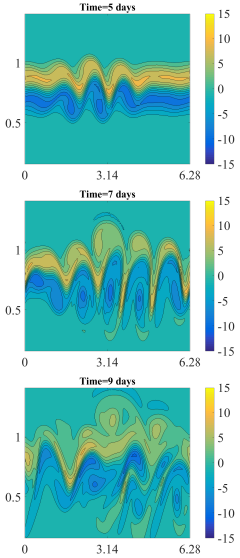
$$\frac{\partial \mathbf{u}}{\partial t} = -(\nabla_n \times \mathbf{u} + f\mathbf{n}) \times \mathbf{u} - \nabla\left(\frac{|\mathbf{u}|^2}{2} + g(h+h_s)\right),$$
$$\frac{\partial h}{\partial t} = -\nabla \cdot (h\mathbf{u}),$$

$\mathbf{u}$ is the velocity, h is the fluid thickness, h_s is the surface level, g is the gravitational acceleration, and f is the Coriolis parameter.

ExpRB methods achieved:

- $\sim 700x$ increase in usable step size over state-of-the-practice IMEX methods,
- $\sim 3x$ increase over previous state-of-the-art exponential RK methods [Gaudreault & Pudykiewicz, 2016].





Multirate reacting flow [R., Gardner, Balos & Woodward, 2019]

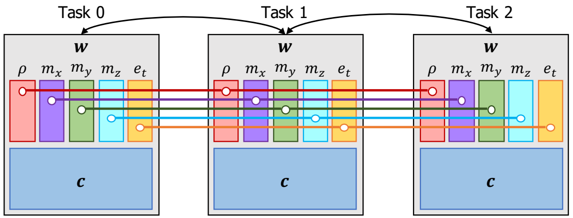
ARKODE demonstration problem: simulates 3D nonlinear compressible Euler equations combined with stiff chemical reactions for a low-density primordial gas, present in models of the early universe.

$$\partial_t \mathbf{w} = -\nabla \cdot \mathbf{F}(\mathbf{w}) + \mathbf{R}(\mathbf{w}) + \mathbf{G}(\mathbf{x},t), \quad \mathbf{w}(t_0) = \mathbf{w}_0,$$

$\mathbf{w}$: density, momenta, total energy, and chemical densities (10)
$\mathbf{F}$: advective fluxes; $\mathbf{R}$: reaction terms; and $\mathbf{G}$: external forces

$\mathbf{w}$ is stored as a *ManyVector*:

- Software layer to treat a collection of vector objects as a single cohesive vector
- Does not touch any vector data directly
- Simplifies partitioning of data among computational resources, e.g., CPUs & GPUs
- May also combine distinct MPI intracommunicators together in a multiphysics simulation.

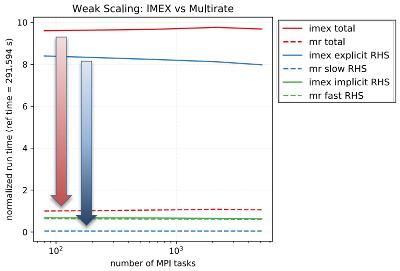


$\mathbf{w}$ is a collection of distributed vectors (density ρ, momentum m_i, and total energy e_T) and local vectors $\mathbf{c}$ (chemical densities).

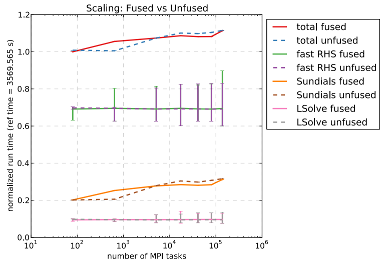
Multirate reacting flow – parallel scalability [R., Gardner, Balos & Woodward, 2019]

Explicit (slow) advection and implicit (fast) reactions; weak scaling with single rate and multirate methods:

- $\mathcal{O}(H^3)$ single rate IMEX method using step sizes set by the reaction time scale
- $\mathcal{O}(H^3)$ 2-rate method with a multirate factor $H/h = 1000$



$\sim 10x$ speedup with multirate

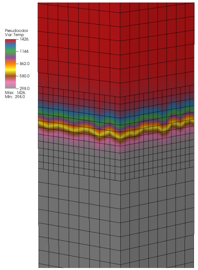


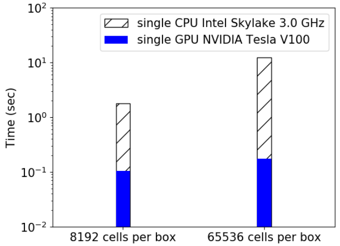
90% weak scaling efficiency using 80 to 138,240 CPU cores of OLCF Summit

Multirating allows advection (which requires MPI) to run at a far larger time step size than that required for the single rate IMEX method to maintain stability, leading to significant speedup.

GPU Reactive Flow Collaborations with AMReX Project – PeleC

PeleC combustion simulation of a perturbed premixed H2-air flame, using ERKSTEP as the fast chemistry integrator in each box of a block-structured AMR mesh:





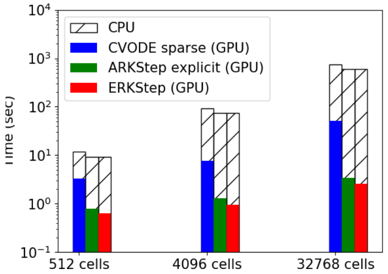
[Graphics courtesy of Hari Sitaraman (NREL); Implementation courtesy of Anne Felden (LBNL) and Hari Sitaraman (NREL)]

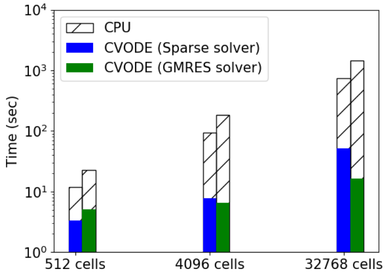
GPU performance compared to a single CPU core:

- 20x faster with 8,192 cells (81,920 ODEs) per box
- 70x faster with 65,536 cells (655,360 ODEs) per box

GPU Reactive Flow Collaborations with AMReX Project – PelePhysics

Simulation of a methane or n-dodecane mixture with oxygen and nitrogen (23 species), comparing SUNDIALS integrators for the chemical network within boxes of 8^3, 16^3, or 32^3 finite volume cells of varying stiffness:





[Graphics courtesy of Hari Sitaraman (NREL); Implementation courtesy of Anne Felden (LBNL) and Hari Sitaraman (NREL)]

- One instance of the integrator is applied to all cells in an AMR box.
- CVODE runs compare matrix-free iterative GMRES vs NVIDIA batched QR linear solvers.
- At 32^3 GPU/CPU speedups are: CVODE sparse ($\sim 15x$), CVODE iterative ($\sim 90x$), ERKSTEP ($\sim 200x$).

Outline

1. Motivation
2. "Flexible" Integrators
3. Applications
4. Conclusions

Conclusions

Large-scale multiphysics problems:

- Nonlinear, interacting models pose key challenges to stable, accurate and scalable simulation.
- Typically large data requirements, requiring scalable/optimal approximation methods.
- While individual physical processes admit 'optimal' algorithms and time scales, these rarely agree.
- Most classical methods invented for idealized problems; perform poorly (or fail) on 'real world' applications.

We aims to develop flexible solvers, that tune the algorithms to the problem (instead of vice-versa), and to implement these in high-quality, open-source software that directly impacts multiphysics applications:

- Method derivation:
  - Stable, accurate and highly efficient multirate methods.
  - Scalable, accurate and practical methods for exponential integrators.
- Software:
  - Support explicit, implicit and IMEX single-rate and multirate methods.
  - Strive for flexibility, enabling user-supplied components that can be optimized for a given problem.