Overview of PETSc Time Stepping Solvers

Presented to
2019 ECP Annual Meeting participants

Hong Zhang
Mathematics and Computer Science Division
Argonne National Laboratory

Royal Sonesta Houston Galleria, Houston, Texas
January 14, 2019

PDEs, Optimization, and Eigenproblems with PETSc/TAO and SLEPc

EXASCALE COMPUTING PROJECT

Argonne NATIONAL LABORATORY
U.S. DEPARTMENT OF ENERGY | Office of Science

Importance of Numerical ODE/DAE Solvers

Needed for almost all time-dependent simulation

- Analytic solutions are rarely of practical use
- Different problems require fundamentally different solution techniques
- Large differences in efficiency depending on the method used

Two main HPC ODE/DAE Solver Packages

- SUNDIALS - Lawrence Livermore National Laboratory
- PETSc - Argonne National Laboratory
- Trilinos - Sandia National Laboratory - has some limited integrators

ODE/DAEs

$$M(t,u)u_t + A(t,u) = B(t,u)$$

$$u(0) = g$$

- $M(t,u)$ - mass matrix
- $A(t,u)$ - stiff portion of equation
- $B(t,u)$ - nonstiff portion

Linear example

$$u_t - Au = 0$$

PETSc TS API

PETSc abstraction

$$\underbrace{M(t,u)u_t + A(t,u)}_{G(t,u,u_t)} = \underbrace{B(t,u)}_{F(t,u)}$$

$$J_\alpha = \alpha G_{u_t} + G_u$$

- User provides
  - **FormRHSFunction**(ts, t, x, F, void *ctx);
  - **FormIFunction**(ts, t, x, xdot, G, void *ctx);
  - **FormIJacobian**(ts, t, x, xdot, alpha, J, Jp, void *ctx);
- The abstraction is general enough to support a wide range of problems
  - $M = \mathbf{I} \Rightarrow$ textbook ODE formulation $u_t = F(t,u)$ and $G_{u_t} = \mathbf{I}$
  - $M = \begin{pmatrix} \mathbf{I} & 0 \\ 0 & 0 \end{pmatrix} \Rightarrow$ Differential Algebraic equation (DAE)
- The Jacobian matrix can be passed down to SNES without extra matrix assembly
- Single step interface so user can have own time loop
- Many holes in the step function for flexibility
  - **TSPreStage(), TSPostStage(), TSPreStep(), TSPostStep(), TSPostEvent()**

PETSc offers a large collection of time integrators

TS Name	Method	Class	Type	Order
euler	forward Euler	one-step	explicit	1
ssp	multistage SSP	Runge-Kutta	explicit	≤ 4
beuler	backward Euler	one-step	implicit	1
cn	Crank-Nicolson	one-step	implicit	2
theta	theta-method	one-step	implicit	≤ 2
alpha(2)	alpha-method	one-step	implicit	2
glle	general linear	general linear	implicit	≤ 3
eimex	extrapolated IMEX	one-step	≥ 1, adaptive	
arkimex	IMEX Runge-Kutta	IMEX Runge-Kutta	IMEX	$1-5$
rosw	Rosenbrock-W	Rosenbrock-W	linearly implicit	$1-4$
glee	method with global error estimation	general linear	explicit/implicit	$1-4$
bdf	standard BDF methods	multistep	implicit	$1-6$
sundials	standard BDF methods	multistep	implicit	variable
basicsymplectic	symplectic methods	one-step	explicit	$1-4$

High higher methods approximate the time derivative with higher order finite differences

- multistep - Use previous solutions (steps) to approximate the time derivatives
- multistage - Use new intermediate solutions (stages) to approximate the time derivatives

Implicit Explicit schemes

Treat the stiff portion of the equation implicitly and the rest explicitly

$$M(t,u^n)\frac{u^{n+1}-u^n}{\Delta t}+A(t,u^{n+1})=B(t,u^n)$$

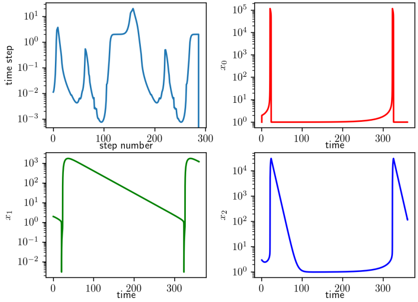
PETSc implements additive Runge-Kutta methods (ARKIMEX)

- Can have L-stable DIRK for stiff part G, SSP explicit part, etc.
- Orders 2 through 5, embedded error estimates
- Dense output, hot starts for Newton
- Extensible adaptive controllers, can change order within a family
- Easy to register new methods: TSARKIMEXRegister()

Rosenbrock schemes

Rosenbrock methods are linearly implicit versions of implicit Runge-Kutta methods. They use explicit function evaluations and implicit linear solves

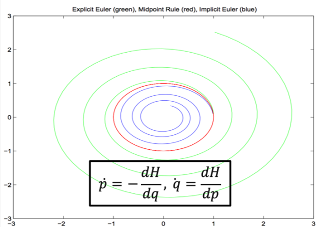
- Faster than the implicit RK because at each stage only a linear system needs to be solved
- Suitable for stiff problems and DAEs and PDAEs
- Approximated Jacobian leads to W-methods



Chemical reaction problem OREGO (TS ex8 OREGO)

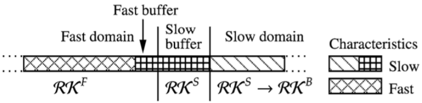
Other partitioned schemes

- Symplectic time integration methods



Separable Hamiltonian system

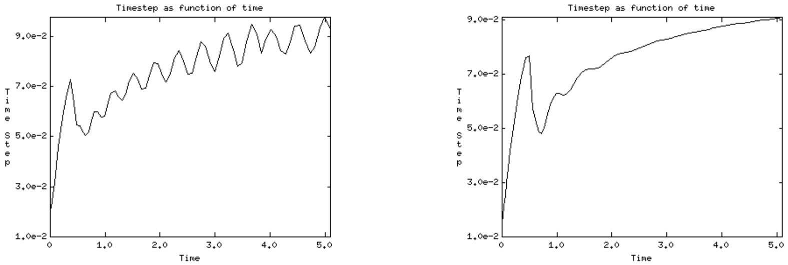
- Partitioned multi-rate Runge-Kutta methods



- Component-wise partitioning based on **TSRHSSplit** (similar to **FieldSplit**)
- Recent new additions, being actively developed

Adaptive Time Stepping

- Estimate the local truncation error induced by the finite differencing in time at each time step by integrating again with a higher order scheme
- Adjust the time-step to keep the local truncation error below a prescribed value
- May decrease or increase the time step
- Can lead to much more efficient (and accurate) computation of the solution
- **Usage** `-ts_adapt_type <basic> | <cfl> | <dsp> | <none>`



basic vs. **DSP** (TS ex11 advection equation)
[courtesy of Lisandro Dalcin]

TSEvent

- Support discontinuous dynamical systems
- Example 1

$$\begin{cases} x - x^{+} = 0, & \text{if } x \geq x^{+} \\ x - x^{-} = 0, & \text{if } x \leq x^{-} \\ \dot{x} = f(x), & \text{otherwise.} \end{cases}$$

- Example 2

$$\begin{cases} \dot{x} = f_1(x), & \text{if } g(t,x) < 0 \\ \dot{x} = f_2(x), & \text{if } g(t,x) \leq 0 \end{cases}$$

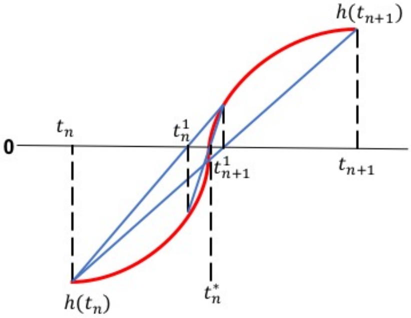
- TS performs integration along with finding “events” – zero crossings of a user-defined function

Event detection and location

- Event detection

$$sign(h(t_n)) \neq sign(h(t_{n+1}))$$

- Event location
  - use interpolation and successively shrink the time boundaries



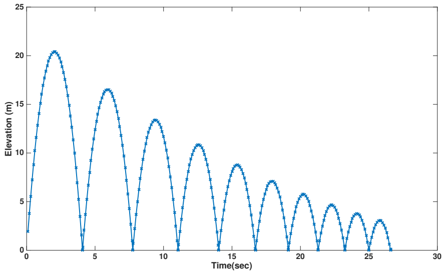
[courtesy of Shrirang Abhyankar]

Usage of TSEvent

```
TSSetEventHandler(TS ts,PetscInt nevents,PetscInt direction[],
  PetscBool terminate[],(*eventfun)(...),(*posteventfun)(...),void *ctx);
```

- nevents – total number of events (can handle multiple event within a timestep)
- direction – type of zero crossing detection
- terminate - terminate on event detection?
- eventmonitor – event monitoring function
- postevent – post-event handling routine

Bouncing ball example(TS ex40)



- Dynamics

$$\begin{cases} u_t = v \\ v_t = -9.8 \end{cases}$$

- Event function $u = 0$
- Post-event function $v = 0.9v$ ($\%10$ reduction in speed)

Takeaways

- PETSc provides a wide variety of high quality, scalable ODE/DAE integrators with different stability and conservation properties
- The TS API supports explicit, implicit and partitioned time integration methods and allows for easy switch between these at runtime
- Adaptive time-stepping provides an inexpensive way to to efficiently integrate ODEs/DAEs
- Event detection and handling enables simulation of hybrid dynamical systems