PETSc and BOUT++

Jed Brown
Peter Brune, Emil Constantinescu,
Debojyoti Ghosh, Lois Curfman McInnes
{jedbrown,brune,emconsta,ghosh,curfman}@mcs.anl.gov

Mathematics and Computer Science Division
Argonne National Laboratory

BOUT++ Workshop, 2013-09-04

Portable **Extensible** Toolkit for Scientific computing

Philosophy: Everything has a plugin architecture

- Vectors, Matrices, Coloring/ordering/partitioning algorithms
- Preconditioners, Krylov accelerators
- Nonlinear solvers, Time integrators
- Spatial discretizations/topology*

Example

Vendor supplies matrix format and associated preconditioner, distributes compiled shared library. Application user loads plugin at runtime, no source code in sight.

Portable Extensible **Toolkit** for Scientific computing

Algorithms, (parallel) debugging aids, low-overhead profiling

Composability

Try new algorithms by choosing from product space and composing existing algorithms (multilevel, domain decomposition, splitting).

Experimentation

- It is not possible to pick the solver *a priori*.
  What will deliver best/competitive performance for a given physics, discretization, architecture, and problem size?
- PETSc's response: expose an algebra of composition so new solvers can be created at runtime.
- Important to keep solvers decoupled from physics and discretization because we also experiment with those.

Outline

Time Integration

Nonlinear solvers

Comments on performance

Trade-offs in time integration

- Properties
  - Nonlinear stability (e.g., positivity preservation)
  - Stability along imaginary axis
  - *L*-stability (damping at infinity)
  - Implicitness and reuse
- What is expensive?
  - Function evaluation
  - Operator assembly/preconditioner setup
    - How much can be reused for how long?
  - Implicit solves
    - Can we find better solver algorithm?
    - More effort in setup?
- What is “convergence”?
  - Wave propagation: implicitness useless for convergence *in a norm*
  - Non-norm functionals could be robust

Reusing implicit solver setup

- Linearization
- MG interpolants
- Lagged preconditioner
- Modified Newton
- Quasi-Newton
- IMEX with linear implicit part
- Rosenbrock/W

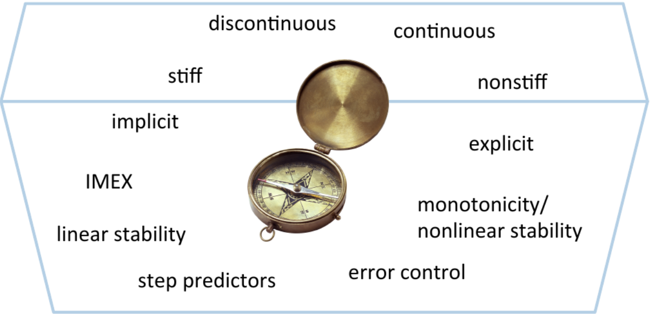
IMEX time integration in PETSc

- Additive Runge-Kutta IMEX methods

$$G(t, x, \dot{x}) = F(t, x)$$
$$J_\alpha = \alpha G_{\dot{x}} + G_x$$

  - User provides:
    - `FormRHSFunction(ts,`t`,`x`,`F`,void *ctx);`
    - `FormIFunction(ts,`t`,`x`,`$\dot{x}$`,`G`,void *ctx);`
    - `FormIJacobian(ts,`t`,`x`,`$\dot{x}$`,`α`,`J`,`J_p`,mstr,void *ctx);`
  - 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
  - More accurate methods if G is linear, also Rosenbrock-W
  - Can use preconditioner from classical “semi-implicit” methods
  - FAS nonlinear solves supported
  - Extensible adaptive controllers, can change order within a family
  - Easy to register new methods: `TSARKIMEXRegister()`
- Single step interface so user can have own time loop
- Same interface for Extrapolation IMEX, LMS IMEX (in development)

Time integration method design



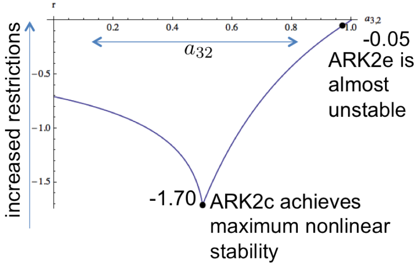
- Select order, number of stages, required properties
- Optimize properties like SSP coefficient, accuracy, or linear stability
- `TSARKIMEXRegister("my-method", ...coefficients...)`
- `-ts_type arkimex -ts_arkimex_type my-method`

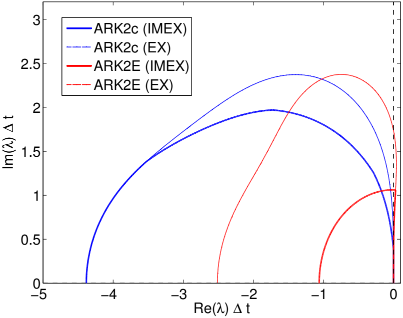
Example: Additive Runge-Kutta design

- 3-stage, second order, *L*-stable implicit part
- one-parameter family of solutions

ARK2c Maximize SSP coefficient

ARK2E Minimize leading error coefficient





Some TS methods

TSSSPRK104 10-stage, fourth order, low-storage, optimal explicit SSP Runge-Kutta $c_{\text{eff}} = 0.6$ (Ketcheson 2008)

TSARKIMEX2E second order, one explicit and two implicit stages, L-stable, optimal (Constantinescu)

TSARKIMEX3 (and 4 and 5), L-stable (Kennedy and Carpenter, 2003)

TSROSWRA3PW three stage, third order, for index-1 PDAE, A-stable, $R(\infty) = 0.73$, second order strongly A-stable embedded method (Rang and Angermann, 2005)

TSROSWRA34PW2 four stage, third order, L-stable, for index 1 PDAE, second order strongly A-stable embedded method (Rang and Angermann, 2005)

TSROSWLLSSP3P4S2C four stage, third order, L-stable implicit, SSP explicit, L-stable embedded method (Constantinescu)

Adaptive controllers

- "Stiff" waves are not stiff if one wants to converge *in a norm*
- PETSc integrators provide embedded methods to estimate errors
- Automatic controllers optimize local truncation error and nonlinear solve cost
- User can register custom controllers
- Use a priori knowledge of the physics, robust functionals
- Choose from list of methods, choose next step size

Outline

Time Integration

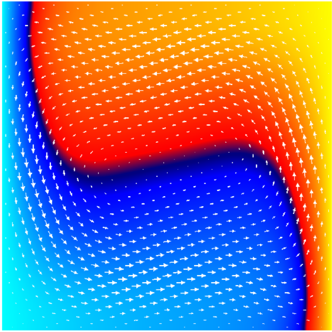
Nonlinear solvers

Comments on performance

Which nonlinear solver?

- Global linearization (NewtonLS, NewtonTR)
  - Preconditioning libraries for assembled matrices
  - Low arithmetic intensity
- Quasi-Newton
  - Build low-rank updates to Jacobian inverse
  - Brown and Brune, "Low-rank quasi-Newton updates for robust Jacobian lagging in Newton-type methods", ANS MC13.
- Nonlinear multigrid and domain decomposition
  - ASPIN (left-preconditioned nonlinear Schwarz), also right-preconditioned
  - Full Approximation Scheme with linear or nonlinear smoothers
  - More intrusive, but freakishly efficient for difficult problems
- Nonlinear GMRES, Anderson mixing, nonlinear CG
  - Accelerator for nonlinear preconditioning
  - Good alternative to matrix-free finite differencing
  - More robust line search possible: operates in reduced basis

- high Rayleigh number (Ra = 2e4) flow
- time, iterations, V-cycles, **intensity** (GFLOPs), MPI **reductions**
- just a **demonstration**; 64 cores, 4k unknowns per core
- Newton-(GMRES-MG) with nonlinear elimination vs. NGMRES-FAS



	NK-MG	NASM*(NK-MG)	NGMRES-FAS
time (sec)	7	4	1
its.	24	12	22
V-Cycles	354	155	22
GFLOPs	11	14	32
MPIReduct	4129	2711	775

The Great Solver Schism: Monolithic or Split?

Monolithic

- Direct solvers
- Coupled Schwarz
- Coupled Neumann-Neumann (need unassembled matrices)
- Coupled multigrid

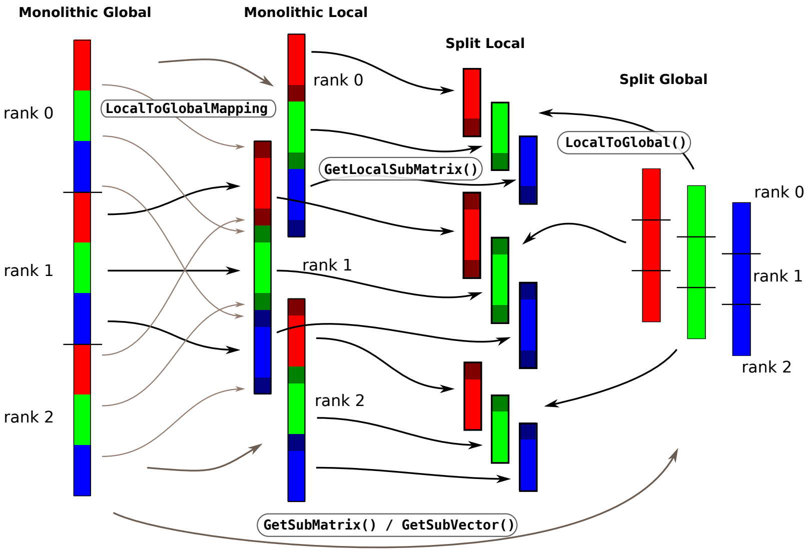
X Need to understand local spectral and compatibility properties of the coupled system

Split

- Physics-split Schwarz (based on relaxation)
- Physics-split Schur (based on factorization)
  - approximate commutators SIMPLE, PCD, LSC
  - segregated smoothers
  - Augmented Lagrangian
  - “parabolization” for stiff waves

X Need to understand global coupling strengths

- Preferred data structures depend on which method is used.
- Interplay with geometric multigrid.



Work in Split Local space, matrix data structures reside in any space.

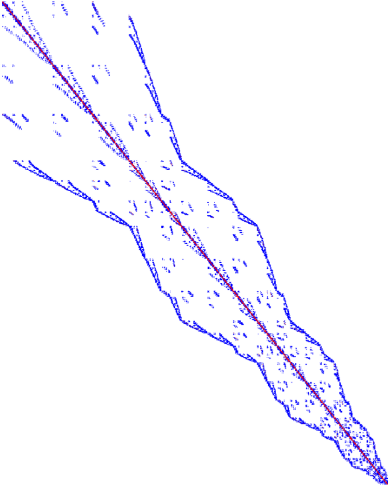
Outline

Time Integration

Nonlinear solvers

Comments on performance

Bottlenecks of (Jacobian-free) Newton-Krylov



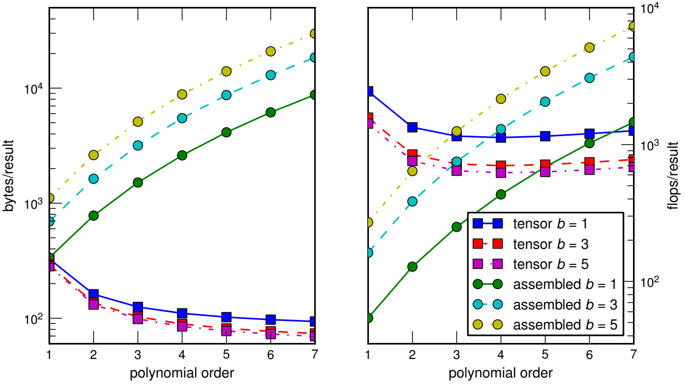
- Matrix assembly
  - integration/fluxes: FPU
  - insertion: memory/branching
- Preconditioner setup
  - coarse level operators
  - overlapping subdomains
  - (incomplete) factorization
- Preconditioner application
  - triangular solves/relaxation: memory
  - coarse levels: network latency
- Matrix multiplication
  - Sparse storage: memory
  - Matrix-free: FPU
- Globalization

Scalability Warning

The easiest way to make software scalable is to make it sequentially inefficient.
(Gropp 1999)

- We really want *efficient* software
- Need a performance model
  - memory bandwidth and latency
  - algorithmically critical operations (e.g. dot products, scatters)
  - floating point unit
- Scalability shows marginal benefit of adding more cores, nothing more
- Constants hidden in the choice of algorithm
- Constants hidden in implementation

Performance of assembled versus unassembled



- High order Jacobian stored unassembled using coefficients at quadrature points, can use local AD
- Choose approximation order at run-time, independent for each field
- Precondition high order using assembled lowest order method
- Implementation $> 70\%$ of FPU peak, SpMV bandwidth wall $< 4\%$

Hardware Arithmetic Intensity

Operation	Arithmetic Intensity (flops/B)
Sparse matrix-vector product	1/6
Dense matrix-vector product	1/4
Unassembled matrix-vector product	≈ 8
High-order residual evaluation	> 5

Processor	BW (GB/s)	Peak (GF/s)	Balanced AI (F/B)
E5-2670 8-core	35	166	4.7
Magny Cours 16-core	49	281	5.7
Blue Gene/Q node	43	205	4.8
Tesla M2090	120	665	5.5
Kepler K20Xm	160	1310	8.2
Xeon Phi	150	1248	8.3