ODE/DAE Integrators and Nonlinear Solvers

Presented to
ATPESC 2018 Participants

Barry Smith
Argonne Distinguished Fellow
Mathematics and Computer Science Division
Argonne National Laboratory
Q Center, St. Charles, IL (USA)
Date 08//06/2018

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

$$M(t,u)u_t + F(t,u) = G(t,u)$$

$$u(0) = g$$

- $M(t,u)$ - mass matrix
- $F(t,u)$ - stiff portion of equation
- $G(t,u)$ - nonstiff portion

Linear example

$$u_t - Au = 0$$

Euler Scheme

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

$$u^{n+1} = u^n + \Delta t G(t, u^n)$$

From PDE to ODE

$$u_t = u_{xx}$$

$$u(0) = u(2\pi) = 0$$

$$u(0, x) = \sum_m \alpha_m \sin(mx)$$

Analytic solution

$$u(t, x) = \sum_m \alpha_m e^{-m^2 t} \sin(mx)$$

From PDE to ODE

Semi-discrete form

$$(u_i)_t = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}$$

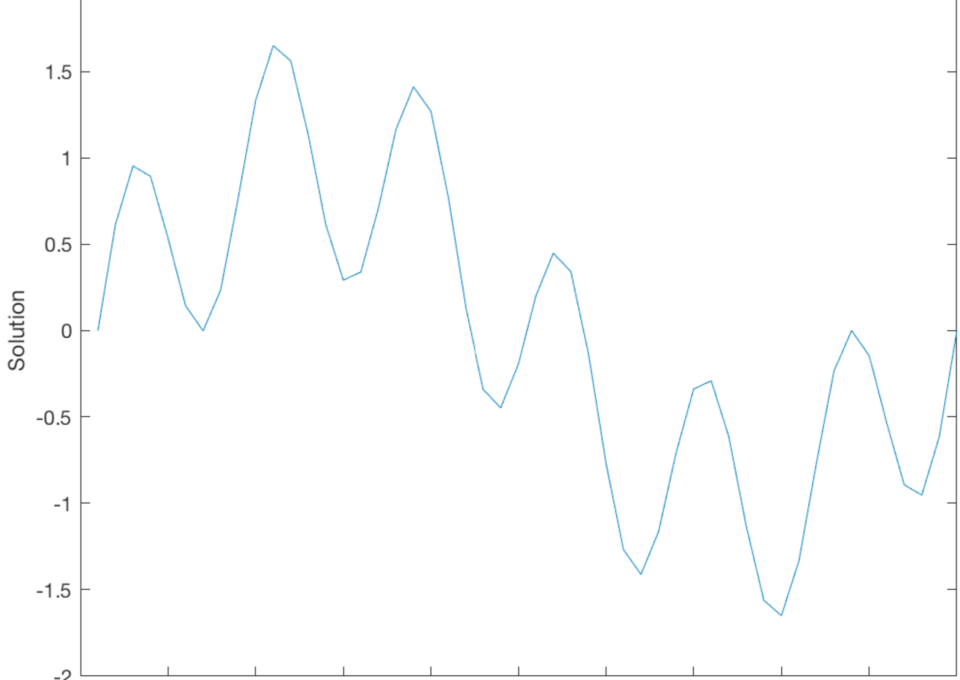
$$u_0 = u_N = 0$$

Fully discrete form with Euler's method

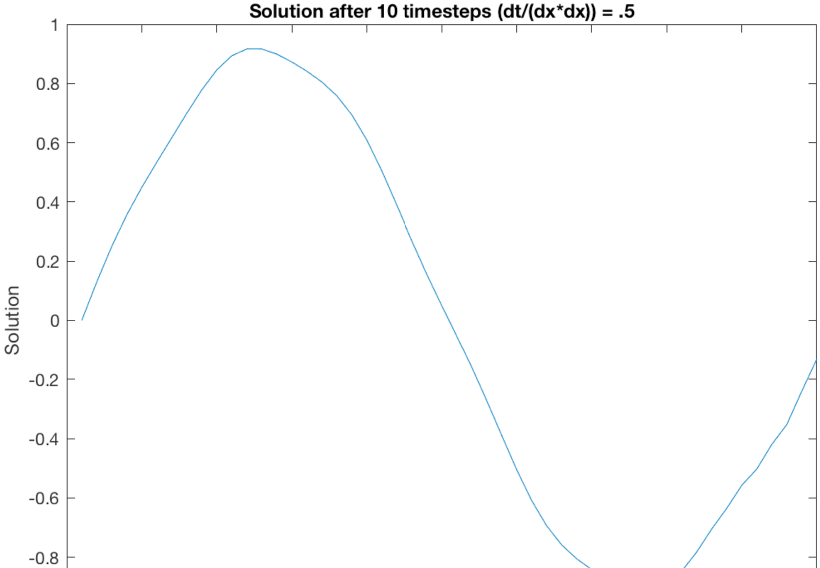
$$u_i^{n+1} = u_i^n + \frac{\Delta t}{(\Delta x)^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

$$u_i^0 = g_i$$

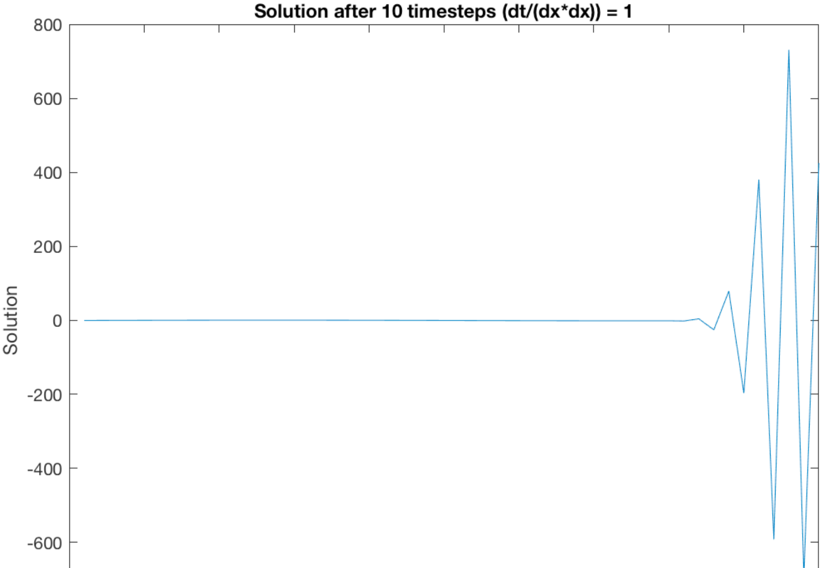
Stab



Stability: Stable Solution



Stability: Unstable Solution



Stability

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{(\Delta x)^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

Consider a solution of the form $u_i^n = \sin(mi\Delta x)$. Does it grow or shrink as we integrate in time?

$$\sin(mi\Delta x) + \frac{\Delta t}{(\Delta x)^2}(\sin(m(i+1)\Delta x) - 2\sin(mi\Delta x) + \sin(m(i-1)\Delta x))$$

$$\sin(mi\Delta x) + \frac{2\Delta t}{(\Delta x)^2}(\sin(mi\Delta x)\cos(m\Delta x) + \cos(mi\Delta x)\sin(m\Delta x) -$$
$$2\sin(mi\Delta x) + \sin(mi\Delta x)\cos(m\Delta x) - \cos(mi\Delta x)\sin(m\Delta x))$$

$$\sin(mi\Delta x) + \frac{2\Delta t}{(\Delta x)^2}\sin(mi\Delta x)(\cos(m\Delta x) - 1)$$

Stability

$$-1 \leq \sin(mi\Delta x)(1 + \frac{2\Delta t}{(\Delta x)^2}(\cos(m\Delta x) - 1)) \leq 1$$

$$-1 \leq 1 + \frac{2\Delta t}{(\Delta x)^2}(-2)$$

$$\frac{\Delta t}{(\Delta x)^2} \leq 1/2.$$

Courant-Friedrichs-Lewy (CFL) condition

Stiffness

- “The stepsize needed to maintain stability of the forward Euler method is much smaller than that required to represent the solution accurately” – U. Ascher and L. Petzold

Backward Euler: Implicit Schemes

$$M(t, u^n)\frac{u^{n+1} - u^n}{\Delta t} + F(t, u^{n+1}) = 0$$

Unconditionally stable, proof, do the same analysis as for Euler and observe that the solution never grows independent of Δt and Δx.

Implicit Explicit Schemes: A Cartoon

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

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

Differential Algebraic Equations (DAE)

$$M(t, u, w)u_t + F(t, u, w) = G(t, u, w)$$

$$H(t, u, w) = 0$$

$$u(0) = g$$

$$w(0) = h$$

Multistep and Multistage Schemes

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

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

Exercise I: Properties of Time Steppings:

https://xsdk-project.github.io/ATPESC2018HandsOnLessons/lessons/time_integrators

Newton's Method:

For implicit methods one needs to solve nonlinear systems

$$Q(w) = 0.$$

From Taylor series

$$Q(w + \delta w) = Q(w) + J_Q(w)\delta w + ... = 0$$

$$\delta w = -J_Q(w)^{-1} Q(w)$$

$$w^{m+1} = w^m - J_Q(w^m)^{-1} Q(w^w)$$

Exercise II: Quadratic and Mesh Independence convergence of Newton's method

PETSc/TAO:

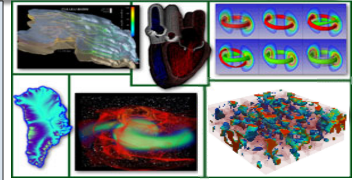
Portable, Extensible Toolkit for Scientific Computation / Toolkit for Advanced Optimization

Scalable algebraic solvers for PDEs. Encapsulate parallelism in high-level objects. Active & supported user community. Full API from Fortran, C/C++, Python.

- Optimization
- Time Integrators
- Nonlinear Algebraic Solvers
- Krylov Subspace Solvers
- Preconditioners
- Domain-Specific Interfaces
  - Networks
  - Quadtree / Octree
  - Unstructured Mesh
  - Structured Mesh
- Vectors | Index Sets | Matrices
- Computation & Communication Kernels

- **Easy customization and composability of solvers at runtime**
  - Enables optimality via flexible combinations of physics, algorithmics, architectures
  - Try new algorithms by composing new/existing algorithms (multilevel, domain decomposition, splitting, etc.)
- **Portability & performance**
  - Largest DOE machines, also clusters, laptops
  - Thousands of users worldwide

Argonne NATIONAL LABORATORY



PETSc provides the backbone of diverse scientific applications. clockwise from upper left: hydrology, cardiology, fusion, multiphase steel, relativistic matter, ice sheet modeling

https://www.mcs.anl.gov/petsc

PETSc: Platform for experimentation

- ***No optimality without interplay among physics, algorithmics, and architectures***
- **Need algebraic solvers to be:**
  - **Composable:** Separately developed solvers may be easily combined, by non-experts, to form a more powerful solver.
  - **Hierarchical:** Outer solvers may iterate over all variables for a global problem, while inner solvers handle smaller subsets of physics, smaller physical subdomains, or coarser meshes.
  - **Nested:** Outer solvers call nested inner solvers.
  - **Extensible:** Users can easily customize/extend.
- Many solver configurations can be set at runtime to avoid needing to recompile.

PETSc/TAO capabilities

Functionality	More Details (Algorithms, Data Structures, etc.)
Optimization	PDE Constrained; Adjoint Based; Derivative Free; Others
Time Integrators	Pseudo-transient; Runge-Kutta; Strong Stability Preserving; Others; General Linear; IMEX; Rosenbrock-W
Nonlinear Algebraic Solvers	Line Search Newton; Quasi-Newton (BFGS); Nonlinear Gauss Seidel; Nonlinear CG; Trust Region Newton; Nonlinear Multigrid (FAS); Successive Substitutions; Active Set VI
Krylov Subspace Solvers	Pipeline methods; GMRES; Chebyshev; BiCG-Stabilized; CG; Hierarchical Krylov; LSQR; SYMMLQ; TFQMR; Others
Preconditioners	Blocks (by field); Additive Schwarz; ILU/ICC; Schur Complement; Algebraic Multigrid; Geometric Multigrid; App-specific; Others
Domain-Specific Interfaces: **Networks**	Infrastructure networks, e.g., electrical, gas, water
Domain-Specific Interfaces: **Quadtree / Octree**	Structured mesh refinement
Domain-Specific Interfaces: **Unstructured Mesh**	Complex domains with finite element and finite volume discretizations
Domain-Specific Interfaces: **Structured Mesh**	Simple domains and discretizations, e.g., finite difference methods
Vectors; **Index Sets**; **Matrices**	Compressed Sparse Row (AIJ); Block AIJ; Matrix Blocks (MatNest); Symmetric Block AIJ; Dense; GPU and Pthread Matrices
Computation & Communication Kernels	MPI, OpenMP, MPI-IO, CUDA, Pthreads, BLAS, LAPACK, etc.

Take Away

- PETSc and SUNDIALS provide a wide variety of high quality, scalable ODE/DAE integrators
- PDEs can be converted to ODEs/DAE via discretization in space and then solved using ODE/DAE libraries
- Stiffness is an important property of ODEs and effects the appropriate schemes to use
- Adaptive time-stepping provides an inexpensive way to to efficiently integrate ODEs/DAEs