

An Examination of Precision Effects on Numerical Solutions of Partial Differential Equations

Paul Rigge[†] and Benson K. Muite^{*}

[†] Department of Electrical Engineering and Computer Science

^{*} Department of Mathematics

University of Michigan

Ann Arbor, MI 48109

October 29-30, 2011

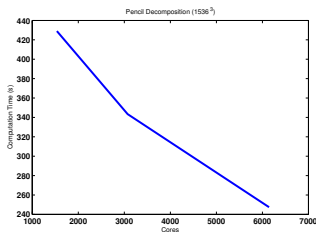
Overview

- 1 Motivation
- 2 Numerical Methods
- 3 Implementation
- 4 Measured Results
- 5 Conclusion

Scaling and Numerical Precision

Problem of interest: Nonlinear Schrödinger Equation

$$iu_t = -\Delta u + u|u|^2$$



Want to use more cores to get an *accurate* result to *large* problem sizes faster. However, roundoff error can accumulate for large problem sizes over long times, leading to inaccurate results.

Sine-Gordon Equation

$$u_{tt} - u_{xx} + \sin u = 0 \quad (1)$$

Factorizes into:

$$u_t + u_x = v \quad (2)$$

$$v_t - v_x = -\sin u \quad (3)$$

Conserved Hamiltonian:

$$\frac{1}{2} (u_t^2 + u_x^2) + 1 - \cos u \quad (4)$$

Numerical Method Overview

- Spectral method
 - spacial derivatives computed with FFTs
- Implicit Runge Kutta for temporal derivatives
 - For every s there exists a IRK method order $2s$
- Fixed point iteration to solve nonlinear part ($\widehat{\sin u} \neq \sin \widehat{u}$)
- Each iteration requires 1 FFT/IFFT pair to recompute nonlinear term
- Each iteration requires solving a $2s \times 2s$ system for s -stage IRK method
- Can simplify this by using a diagonally implicit method, then only system of $2s$ independent equations

Spectral Methods

Spectral methods use Discrete Fourier Transforms (DFTs) to approximate derivatives. The DFT and inverse DFT are given by

$$\hat{v}_k = h \sum_{j=1}^N e^{-ikx_j} v_j, \quad k = -\frac{N}{2} + 1, \dots, \frac{N}{2} \quad (5)$$

$$v_j = \frac{1}{2\pi} \sum_{k=-N/2+1}^{N/2} e^{ikx_j} \hat{v}_k, \quad j = 1, \dots, N \quad (6)$$

- For smooth functions, Fourier coefficients converge faster than any polynomial.
- Can be computed in $O(n \log n)$ operations with Fast Fourier Transform (FFT)
- Given a function u with Fourier coefficients \hat{u}_k , the derivative $\frac{\partial u}{\partial x}$ has Fourier coefficients $ik\hat{u}_k$.

Runge-Kutta Methods

Given an evolution equation of the form $u_t = f(u)$, a Runge-Kutta method for u is

$$U_i = u^n + h \sum_{j=1}^s a_{ij} f(U_j) \quad (7)$$

$$u^{n+1} = u^n + h \sum_{i=1}^s b_i f(U_i) \quad (8)$$

where h is the timestep, $\mathbf{A} = (a_{ij})$ and $\mathbf{b} = (b_i)$ are the Runge-Kutta coefficients, and U_i are the s intermediate stage variables.

- If $a_{ij} = 0$ for $j \geq i$, the method is explicit, implicit otherwise.
- Important fact: Runge-Kutta methods based on Gauss-Legendre quadrature have convergence $2s$, exist for all s (implicit)

IRK Methods for sine-Gordon

An s -stage IRK scheme for sine-Gordon is

$$\hat{U}_i = \hat{u}^n + \Delta t \sum_{j=1}^s a_{ij} \left(-ik\hat{U}_j + \hat{V}_j \right), \quad i = 1, \dots, s \quad (9)$$

$$\hat{V}_i = \hat{v}^n + \Delta t \sum_{j=1}^s a_{ij} \left(ik\hat{V}_j - \widehat{\sin(U_j)} \right), \quad i = 1, \dots, s \quad (10)$$

$$\hat{u}^{n+1} = \hat{u}^n + \Delta t \sum_{i=1}^s b_i \left(-ik\hat{U}_i + \hat{V}_i \right) \quad (11)$$

$$\hat{v}^{n+1} = \hat{v}^n + \Delta t \sum_{i=1}^s b_i \left(ik\hat{V}_i - \widehat{\sin(U_i)} \right) \quad (12)$$

Fully Implicit

$$\left(\mathbf{I} + \Delta t k i \frac{\mathbf{A}}{-\mathbf{A}} + \Delta t \frac{\mathbf{A}}{-\mathbf{A}} \right) \begin{pmatrix} \hat{U}' \\ \hat{V}' \end{pmatrix} = \begin{pmatrix} \hat{u}^n \mathbf{e} \\ \hat{v}^n \mathbf{e} - dt \mathbf{A} \sin(\widehat{U'^{-1}}) \end{pmatrix} \quad (13)$$

- \hat{U}' is the next iteration, \hat{U}'^{-1} is the previous iteration
- Must solve $2s \times 2s$ system at each wavenumber, every iteration
- Our FORTRAN implementation prefactorizes LHS matrix-
 $O(N_x s^2)$ space

Diagonally Implicit

$$(1 + kia_{ii})\hat{U}'_i = \hat{u}^n + \Delta t \sum_{j=1}^s -ik\tilde{a}_{ij}\hat{U}'_j{}^{-1} + a_{ij}\hat{V}'_j{}^{-1}, \quad i = 1, \dots, s \quad (14)$$

$$(1 + kia_{ii})\hat{V}'_i = \hat{v}^n + \Delta t \sum_{j=1}^s ik\tilde{a}_{ij}\hat{V}'_j{}^{-1} - a_{ij}\widehat{\sin(U'_j{}^{-1})}, \quad i = 1, \dots, s \quad (15)$$

- $\tilde{a}_{ij} = a_{ij}$ if $i \neq j$, 0 otherwise
- Off-diagonal linear terms and nonlinear term solved iteratively

Some Implementation Details

- Compilers: ifort, gfortran
- Netlib lapack (modified for quad precision operations)
- Ooura's FFT Library (not particularly tuned, also modified for quad precision)
- Python script to generate arbitrary precision IRK coefficients for any number of stages

IRK Coefficients

Algorithm:

- ① Find approximate values (double precision) for the roots of the s -degree Legendre polynomial
- ② Find higher-precision values for each root with arbitrary precision iterative solver
- ③ $c_i = \frac{1}{2}(r_i + 1)$ for every root r_i
- ④ For each $i = 1, \dots, s$, $a_{i,j}$ is the solution to $\sum_{j=1}^s a_{ij} c_j^{k-1} = \frac{1}{k} c_i^k$ for $k = 1, \dots, s$
- ⑤ For each $j = 1, \dots, s$ find b_j as the solution to $\sum_{j=1}^s b_j c_j^{k-1} = \frac{1}{k}$ for $k = 1, \dots, s$

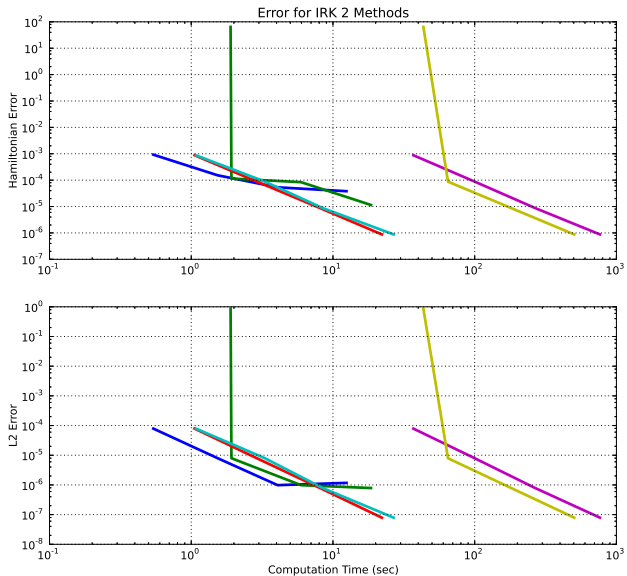
Compute in higher precision than the final precision of the coefficients. If it is computed with the same precision, accumulated roundoff will lead to some error and codes using these coefficients will not converge to machine precision.

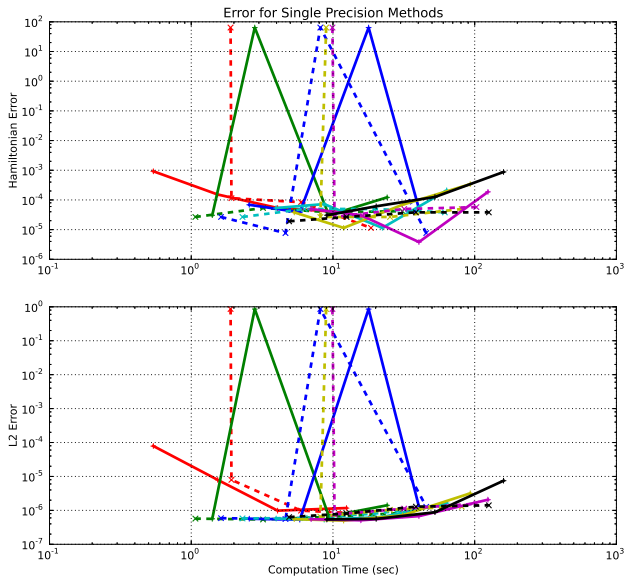
Test Solution

$$u = -4 \tan^{-1} \left(\frac{m}{\sqrt{m^2 - 1}} \frac{\sinh(t\sqrt{m^2 - 1})}{\cosh(mx)} \right) \quad (16)$$

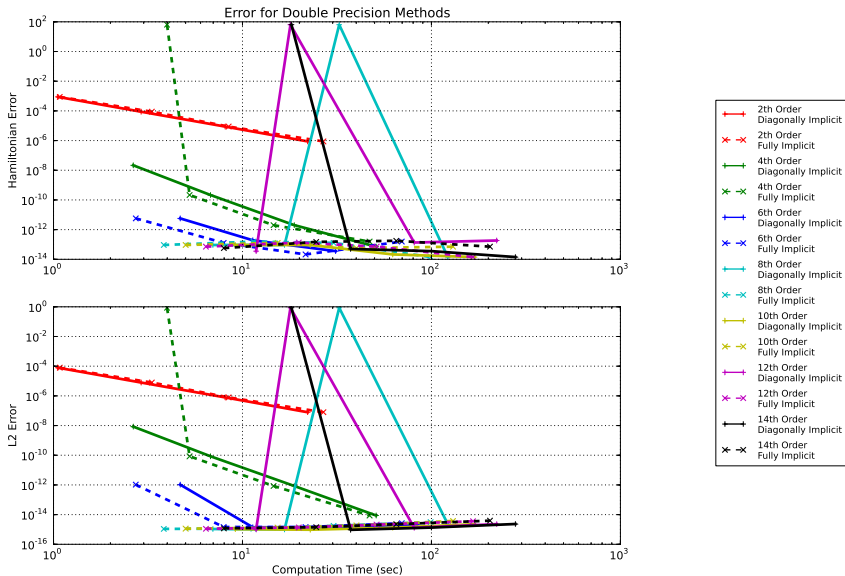
- Plotted for $m = 4$ on $x \in [-4\pi, 4\pi]$.
- Fixed point iteration tolerance = $100\epsilon(h)$
- Hamiltonian, L2 error computed at end time ($t = 1$)

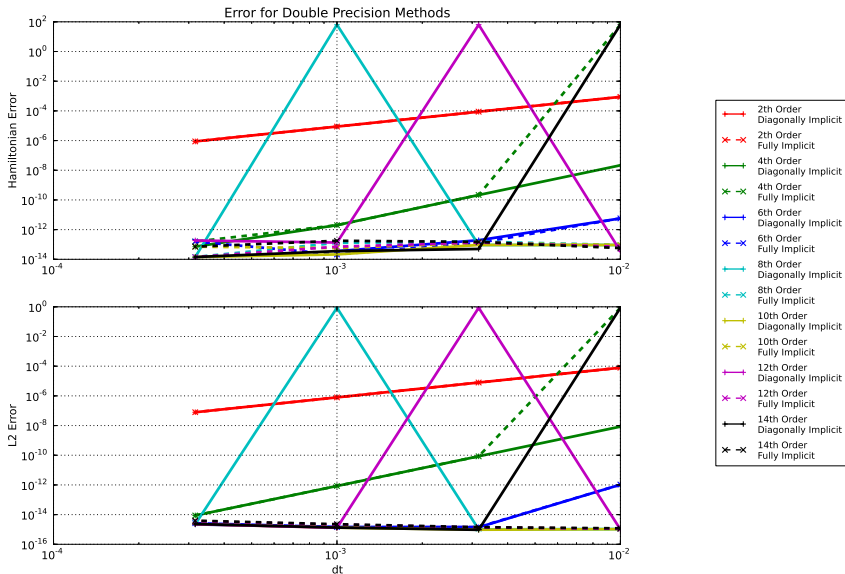
(Loading Video...)

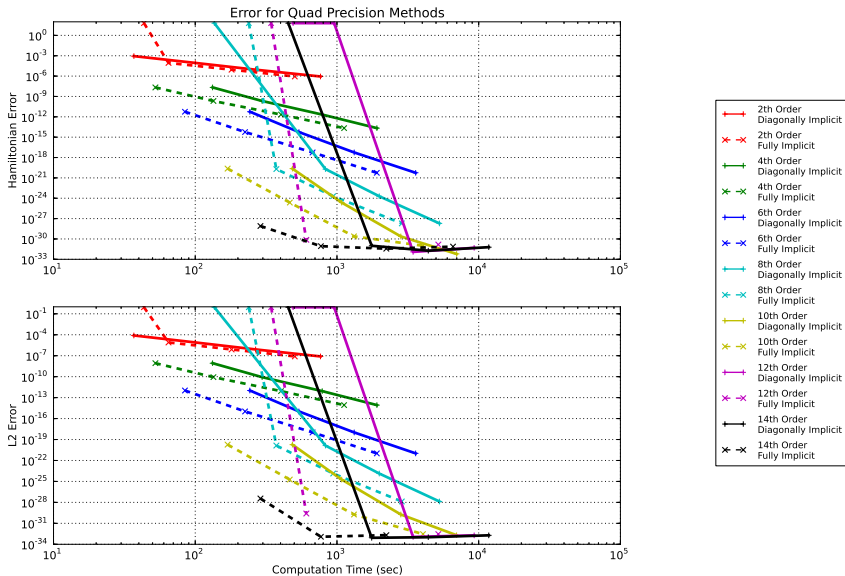


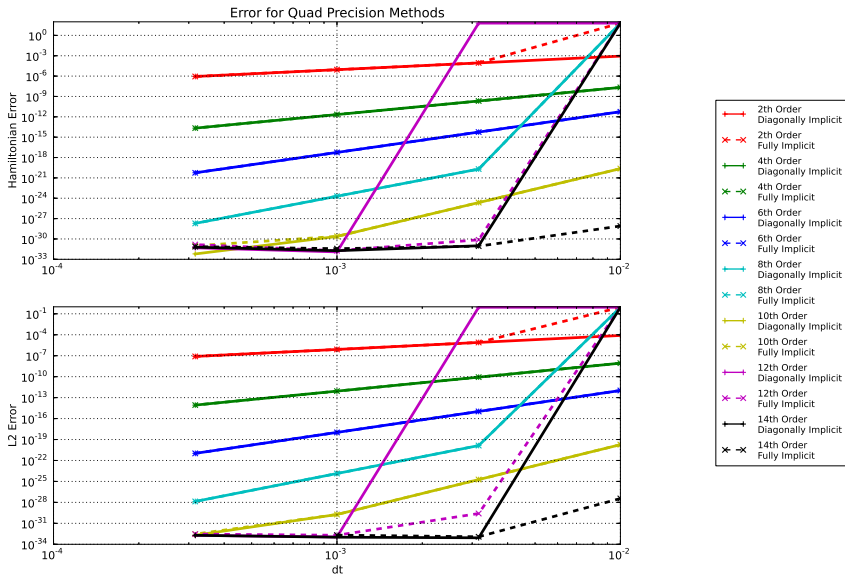


- 2th Order Diagonally Implicit
- x- 2th Order Fully Implicit
- 4th Order Diagonally Implicit
- x- 4th Order Fully Implicit
- 6th Order Diagonally Implicit
- x- 6th Order Fully Implicit
- 8th Order Diagonally Implicit
- x- 8th Order Fully Implicit
- 10th Order Diagonally Implicit
- x- 10th Order Fully Implicit
- 12th Order Diagonally Implicit
- x- 12th Order Fully Implicit
- 14th Order Diagonally Implicit
- x- 14th Order Fully Implicit









Homoclinic Orbit

Exact solution

$$u(x, t) = \pi + 4 \tan^{-1} \left(\frac{\tan \nu \cos[(\cos \nu)x]}{\cosh[(\sin \nu)t]} \right).$$

- Unstable equilibrium, interesting benchmark for accuracy of numerical scheme

(Loading Video...)





Summary

- Single precision did not offer a significant speedup versus double precision (may be untrue for SIMD architectures)
- Penalty of $100\times$ moving a method from double to quad precision (hardware+compiler dependent)
- 8th order method best for double precision
- 14th order method best so far for quad precision
- Fully implicit methods more stable than diagonally implicit, also faster
- Using higher numerical precision allows unstable solutions to be followed longer

Further Work

- Examine convergence of iterative methods.
- Parallel computations for nonlinear Schrödinger equation using Runge-Kutta methods.
- Look at even higher order methods for quad precision

References

-  Butcher, J. (1964). Implicit Runge-Kutta processes. *Math. Comp.* 18(85), 50-64
-  Ercolani, N., M. Forest, and D. McLaughlin (1990). Geometry of the modulational instability III. Homoclinic orbits for the periodic sine-Gordon equation. *Physica D* 43, 349-84.
-  Leimkuhler, B. and S. Reich (2004). *Simulating Hamiltonian Dynamics*. Cambridge University Press.
-  Trefethen, L. (2000). *Spectral Methods in MATLAB*. SIAM.

Acknowledgements

- Funding for this work was provided by the National Science Foundation's Office of CyberInfrastructure through the Blue Waters Undergraduate Petascale Education Program.
- We also thank Ning Li, Christian Klein and Peter Miller for helpful discussions.