Precision Effects on Numerical Solutions to the Sine-Gordon Equation

P. Rigge†, B.K. Muite⋆
riggep@umich.edu, muite@umich.edu

†*Department of Electrical Engineering and Computer Science*, ⋆*Department of Mathematics*
University of Michigan, Ann Arbor

Abstract

We examine numerical precision effects for the Sine-Gordon equation. We implement high order implicit Runge Kutta solvers using fixed-point iteration and compare diagonally and fully implicit schemes. We find that in quadruple precision, fourteenth order time stepping schemes are very efficient.

Introduction

The sine-Gordon Equation is given by $u_{tt} - u_{xx} + \sin u = 0$ and factorizes into:

$$u_t + u_x = v \qquad\qquad v_t - v_x = -\sin u$$

Spectral Methods

Spectral methods use Discrete Fourier Transforms (DFTs) to approximate derivatives. The DFT is given by:

$$\hat{v}_k = h \sum_{j=1}^{N} e^{-ikx_j} v_j,\ k = -\frac{N}{2}+1, \ldots, \frac{N}{2}$$

Given a function u with Fourier coefficients $\hat{u}_k$, the derivative $\frac{\partial u}{\partial x}$ has Fourier coefficients $ik\hat{u}_k$. For smooth functions, Fourier coefficients converge faster than any polynomial, so approximating derivatives is very accurate with a relatively small number of grid points. The DFT can be computed efficiently in $O(n \log n)$ operations with the Fast Fourier Transform (FFT).

Runge-Kutta Methods

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

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

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. Implicit Runge-Kutta methods based on Gauss-Legendre quadrature have convergence $2s$ and exist for all s. If $f(u)$ is nonlinear, as is the case for the sine-Gordon equation, we must iteratively solve a nonlinear system of equations. We use two approaches: a fully implicit method and a diagonally implicit method.

Fully Implicit

$$\left(\mathbf{I} + \Delta t k i \left[\begin{array}{c|c} \mathbf{A} & \\ \hline & -\mathbf{A} \end{array}\right] + \Delta t \left[\begin{array}{c|c} & \mathbf{A} \\ \hline \mathbf{A} & \end{array}\right]\right) \begin{pmatrix} \hat{U}' \\ \hat{V}' \end{pmatrix} = \begin{pmatrix} \hat{u}^n \mathbf{e} \\ \hat{v}^n \mathbf{e} - dt \mathbf{A} \widehat{\sin(U)} \end{pmatrix}$$

U and V are vectors of the s intermediate variables, $\hat{U}'$ and $\hat{V}'$ are the next iterations. This method must solve a $2s \times 2s$ system at each wavenumber for every iteration.

Diagonally Implicit

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

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

where $\tilde{a}_{ij} = a_{ij}$ if $i \neq j$, 0 otherwise. The off-diagonal linear terms and nonlinear term are both solved iteratively.

Implementation Details

The solver was written in FORTRAN using Intel and GNU compilers. We used Netlib lapack and Ooura's FFT Library, which were both modified to also use quad precision. We also used a python script to generate the IRK coefficients.

Measured Results

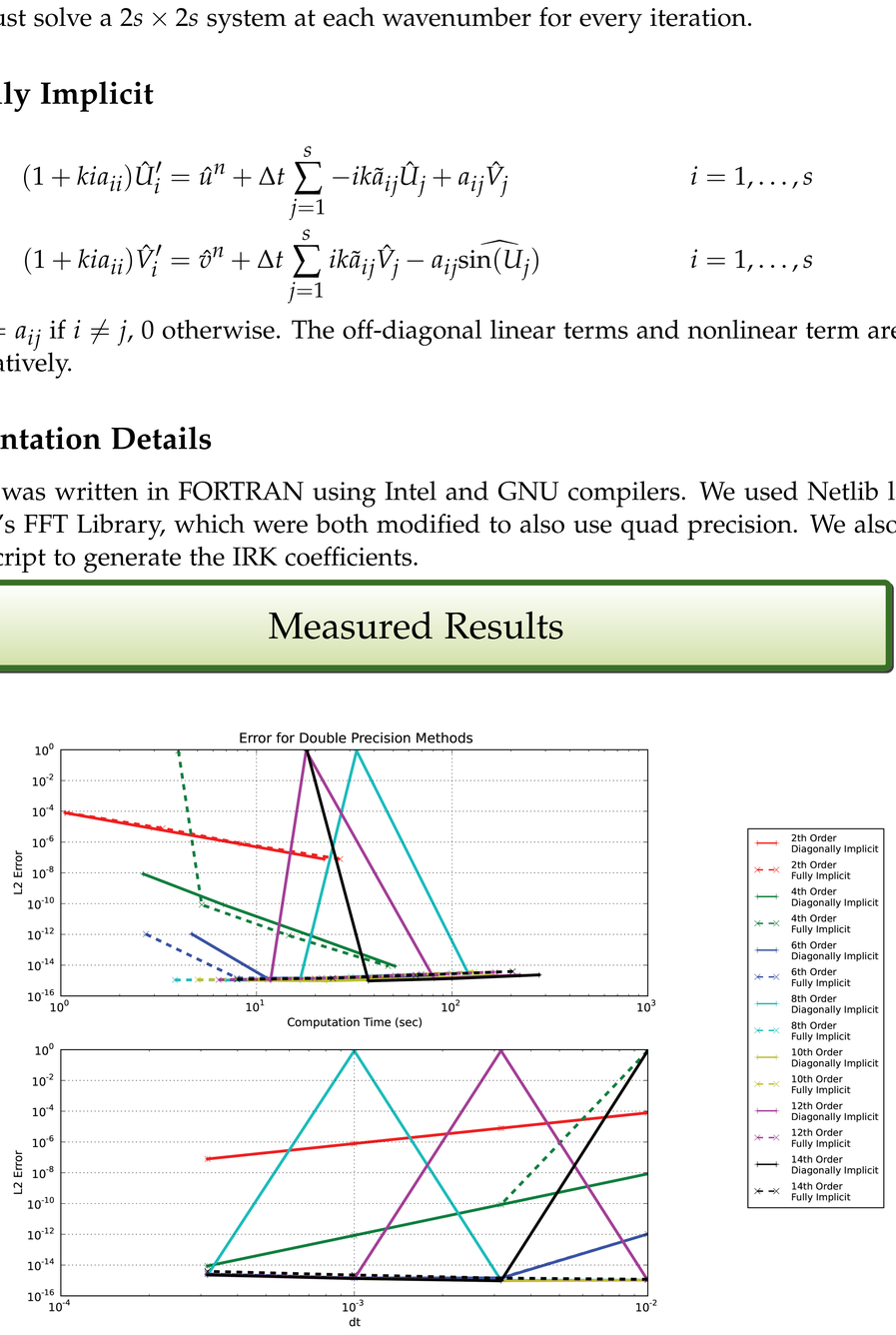

FIGURE 1: Diagonally implicit methods have some problems converging. Fully implicit 8th order appears to be the best method.

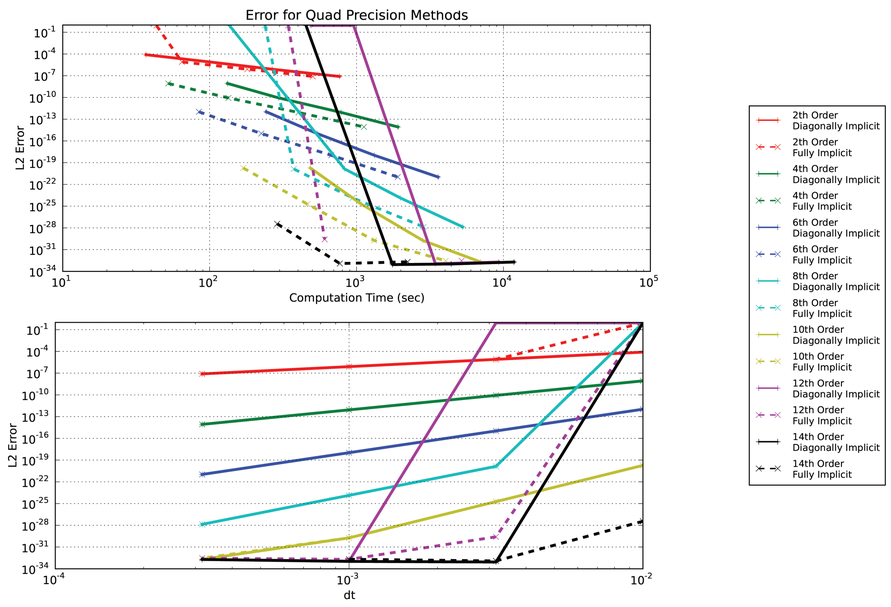

FIGURE 2: Again, diagonally implicit methods have some problems converging. Fully implicit 14th order appears to be the best method.

The test solution is $u = -4\tan^{-1}\left(\frac{m}{\sqrt{m^2-1}}\frac{\sinh(t\sqrt{m^2-1})}{\cosh(mx)}\right)$ plotted for $m = 4$ on $x \in [-4\pi, 4\pi]$. The fixed point iteration tolerance is 100 times the machine precision of the timestep, and the errors were computed at $t = 1$.

Homoclinic Orbit

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

This solution tends to an unstable equilibrium of the equation at $u(x, t \to \infty) = \pi$, which is an interesting benchmark for accuracy of numerical schemes.

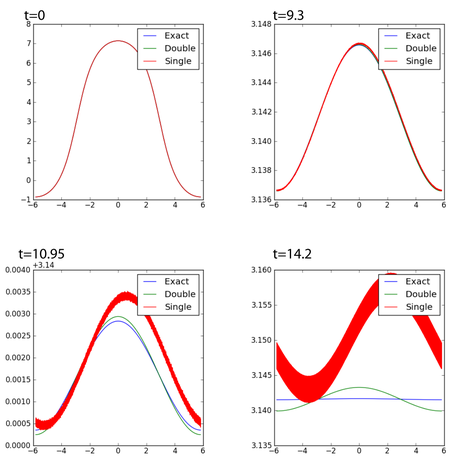

FIGURE 3: Both single and double precision solvers track the exact solution for some time, before the single precision loses it around $t = 9.3$. Some time later, the double precision solver also looses the exact solution. By $t = 14.2$, neither solver resembles the exact solution.

Conclusion

- Penalty of 100× 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

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 Christian Klein and Peter Miller for helpful discussions.

References

[1] Butcher, J. (1964). Implicit Runge-Kutta processes. *Math. Comp. 18*(85), 50-64

[2] 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.

[3] Leimkuhler, B. and S. Reich (2004). *Simulating Hamiltonian Dynamics*. Cambridge University Press.

[4] Trefethen, L. (2000). *Spectral Methods in MATLAB*. SIAM.