

237921

UCRL-ID-139138

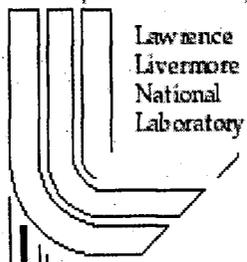
Numerical Errors in DNS: Total Run-Time Error

L.M. Jameson

June 6, 2000



U.S. Department of Energy



Lawrence
Livermore
National
Laboratory

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Work performed under the auspices of the U. S. Department of Energy by the University of California Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

This report has been reproduced
directly from the best available copy.

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information
P.O. Box 62, Oak Ridge, TN 37831
Prices available from (423) 576-8401
<http://apollo.osti.gov/bridge/>

Available to the public from the
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd.,
Springfield, VA 22161
<http://www.ntis.gov/>

OR

Lawrence Livermore National Laboratory
Technical Information Department's Digital Library
<http://www.llnl.gov/tid/Library.html>

Numerical Errors in DNS: Total Run-Time Error

Leland Jameson

Abstract

Understanding numerical errors in simulations is critical for many reasons. First and foremost, one must have some estimate concerning the reliability of the final result. Simply put, numerical errors add up over time and in most cases the increase is a linear process. It is quite possible that running a code for a very long time can lead to a solution which is completely meaningless even though it may look reasonable. This manuscript will begin a technical discussion on these issues.

Contents

1	Introduction	3
2	Generating Difference Equations	4
2.1	Setting up a Linear System	4
2.2	Interpolation	5
2.2.1	Algebraic Polynomials	6
2.2.2	Trigonometric Polynomials	7
3	Error Bounds for Explicit Spatial Operators	8
3.1	Bounds on Differentiation	9
4	Total Error for Space and Time	10
4.1	Runge-Kutta Schemes	10
4.2	The Runge-Kutta Order "Misnomer"	11
4.3	Balancing Temporal and Spatial Errors	12
5	Conclusion	13

List of Figures

1 Introduction

To begin, we must make a few statements which are obvious to some and completely unknown to others. First of all, numerical error do build up over time and never go away, entropy is non-decreasing. More precisely, if the build up is linear, as it usually is, then one can expect a final run-time error to be,

$$Error_{final} = (Error_{per\ time\ step}) * (Number\ of\ Time\ Steps). \quad (1)$$

This very simple expression should always be kept in mind in order to do back of the envelop estimates. To illustrate, a first order scheme on a grid of 1024 by 1024 by 1024 will have a per time step error on the order of 10^{-3} 10^{-4} , depending on the details of the scheme. This means that one can expect to be able to execute only about 1000 time steps before the error is of the same order as the fields involved. This would be a worse case scenario.

In this manuscript we will explore truncation error of commonly occurring finite difference schemes and Runge-Kutta time advancement operators.

2 Generating Difference Equations

Given a vector of N numbers \vec{f} how can we get an approximate value of the derivative \vec{f}' at the i -th point and how good will this approximate value be. Generally speaking, the more elements around the i -th point of \vec{f} that are used to approximate \vec{f}' the better the approximation will be. Common finite difference formulas are found by fitting an algebraic polynomial of degree q locally around the i -th point of a vector \vec{f} of evenly-spaced elements to obtain difference approximations of accuracy $q - 1$. This section will generalize this concept to find the difference equations of arbitrary accuracy on arbitrary grids using algebraic, trigonometric, cosine and exponential polynomials. As special cases, one can obtain all the usual finite difference formulas as well as the Fourier collocation and Chebyshev collocation spectral differential matrices.

Two methods of generating the differencing coefficients will be introduced. The first method explains how to set up a system of equations which will have as a solution the differencing coefficients. The second method is the derivation of differencing coefficients by interpolation. It is this second method which is used throughout the paper for the actual generation of difference equations.

2.1 Setting up a Linear System

The problem is to find a set of coefficients $\{r_k\}$ which combines the raw data in a vector \vec{f} to provide an approximation to a derivative:

$$f'(x_j) = \sum_{k=left}^{k=right} r_k f(x_k). \quad (2)$$

If we require the above equation to be exact for polynomials, algebraic, trigonometric, cosine or exponential, then a linear system of equations can be solved to find an appropriate set of differencing coefficients $\{r_k\}$. Let $b(x)$ denote a fundamental basis element from which a basis can be generated by taking powers of $b(x)$: $b(x) = x$, $b(x) = e^{ix}$, $b(x) = \cos(x)$, or $b(x) = e^x$. That is, we require that the derivative be exact up to a given order N on the numerical grid. The system of equations to be solved for a centered differentiation stencil is

as follows:

$$n(b(x_j))^{n-1}b'(x_j) = \sum_{k=-L}^L r_k(b(x_{j+k}))^n, \quad (3)$$

From this equation one can generate a system of equations with N requirements, that N functions be differentiated exactly, and N degrees of freedom, the N differencing coefficients r_k . If one is near a boundary, then the stenciled is biased. Since this type of system is well-known for algebraic polynomials, an example for the less well-known trigonometric polynomials will be given.

Consider a trigonometric polynomial on a 3 point centered stencil. The first equation simply requires that the derivative of a constant be zero:

$$0 = r_{-1} + r_0 + r_1. \quad (4)$$

Note that is the same equation as for algebraic polynomials since $(x)^0 = (e^{ix})^0$. The next two equations come from requiring that the $n = 1$ mode is differentiated exactly:

$$ie^{ix_0} = r_{-1}e^{ix_{-1}} + r_0e^{ix_0} + r_1e^{ix_1}. \quad (5)$$

One now obtains the two equations from equating the real and imaginary parts. These three equations can be solved for the three coefficients r_{-1}, r_0, r_1 .

Similarly, one can find the coefficients for higher order schemes by requiring that more modes be differentiated exactly. Note that no restrictions were placed on the grid. Differencing formulas can be found on arbitrary grids as easily as they can be found on uniform grids. Also, note that the Fourier spectral differentiation matrix can be found from the above procedure by requiring that the grid be uniform and that the differencing formulas have maximum accuracy on a given grid. That is, if one is working on a grid of size 33 then require that the first 16 modes and the zero-th mode are differentiated exactly.

2.2 Interpolation

A second approach, and the one used in this paper, is to generate differencing coefficients by first interpolating a polynomial through a set of data, followed by differentiation of this polynomial and evaluated at a grid point.

The main reason that differentiation was studied with a variety of types of differentiation operators was to find out if there was any advantage to using, say, trigonometric polynomials to differentiate as opposed to algebraic polynomials when the function to be differentiated was for example a Gaussian pulse. It seemed like an appropriate study to undertake given the current research activity in the area of aeroacoustics where one is often confronted with the need to computationally propagate some type of wave motion. The thought was that perhaps trigonometric polynomials might have some advantage at propagating wave motion over the more common algebraic polynomials. One of the conclusions of this section is that there is no advantage and that one should simply use algebraic polynomials for the generation of differencing equations. In fact, the only important issues involved with obtaining approximate derivatives is the order of the finite difference operator and the density of the numerical grid.

The most important reference for this section is [3]. The following four subsections will cite the interpolation formulas for the four types of interpolation, and hence differentiation, considered in this section.

2.2.1 Algebraic Polynomials

Interpolation with algebraic polynomials is probably the most common form of interpolation, and it is from this type of interpolation that common uniform grid finite difference methods can be found. Using the following formula one can find the finite difference coefficients for an arbitrary grid and of arbitrary order. One simply fits the polynomial to the data, followed by differentiation of the polynomial, and finally one evaluates the polynomial at the point of interest. The well-known Lagrange interpolation formula for algebraic interpolation is,

$$A_j(x) = \prod_{k=0, k \neq j}^n (x - x_k) / \prod_{k=0, k \neq j}^n (x_j - x_k). \quad (6)$$

$A_j(x_k) = \delta_{jk}$ For given values w_0, w_1, \dots, w_n , the polynomial

$$p_n(x) = \sum_{k=0}^n w_k A_k(x). \quad (7)$$

in P_n and takes on these values at the points x_i :

$$p_n(x_k) = w_k, \quad (8)$$

for $k = 0, 1, \dots, n$.

2.2.2 Trigonometric Polynomials

As seen from the previous section, one can also generate difference operators by using trigonometric functions as the fundamental interpolation elements. The following is the appropriate Lagrange-type interpolation formula, see [3]:

For $-\pi \leq x_0 < x_1 < \dots < x_{2n} < \pi$ then

$$T_j(x) = \prod_{k=0, k \neq j}^{2n} \sin \frac{1}{2}(x - x_k) / \prod_{k=0, k \neq j}^{2n} \sin \frac{1}{2}(x_j - x_k). \quad (9)$$

The function,

$$T(x) = \sum_{k=0}^{2n} w_k T_k(x) \quad (10)$$

is the unique solution of the interpolation problem,

$$T(x_k) = w_k, \quad (11)$$

for $k = 0, 1, \dots, 2n$. Again, one can derive finite difference coefficients by interpolating to a function, followed by differentiation of the interpolation polynomial and evaluation at the point of interest. The following section will prove that such difference equations obey order properties just as the usual difference equations derived from algebraic polynomials do.

3 Error Bounds for Explicit Spatial Operators

Here we will limit our discussion to explicit finite difference operators. First we ask the question what are bounds for high order polynomial interpolation.

Given a function $f(x)$ we would like to approximate it with a high order algebraic polynomial $P_N(x)$. As we have seen above, the truncation error is,

$$|P_N(x) - f(x)| = \frac{f^N(\xi)}{N!} \prod_{k=1}^N (x - x_k) \quad (12)$$

Now, if we are approximating trigonometric polynomials, $f(x) = e^{ikx}$, then this truncation is bounded by,

$$|P_N(x) - f(x)| \leq \frac{k^N}{N!} \prod_{k=1}^N (x - x_k), \quad (13)$$

since $f'(x) = ik e^{ikx}$. For the purposes of building centered numerical differentiation operators we are interested in the value of this truncation error when evaluated in the middle of the stencil. We assume that our stencil is composed of an even number of points. Evaluation of

$$\prod_{k=1}^N (x - x_k)$$

in the middle of the stencil leads to the product of the distance between the point of evaluation and each of the stencil points yielding a product such as,

$$\prod_{k=1}^N (x - x_k) = (h/2)^N \prod_{k=1}^{N/2} (2k - 1)^2 \quad (14)$$

so that the interpolation error when evaluating in the middle is bounded by,

$$|P_N(x) - f(x)| \leq \frac{k^N}{N!} (h/2)^N \prod_{k=1}^{N/2} (2k - 1)^2. \quad (15)$$

3.1 Bounds on Differentiation

Now, if we differentiate this polynomial and evaluate at the middle grid point we get the error in the derivative is given by,

$$Err = h^N ((N/2)!)^2 \frac{1}{(N+1)!} f^{(N+1)}(\xi) \quad (16)$$

and when applied to trigonometric functions we get,

$$Err \leq h^N ((N/2)!)^2 \frac{1}{(N+1)!} k^{N+1} \quad (17)$$

where we recall that h is the distance between grid points, N is the order of the approximation, and k is the wavenumber of a Fourier expansion.

4 Total Error for Space and Time

The error in a numerical calculation will depend on both the spatial and temporal discretization. Here we consider one common temporal discretization, Runge-Kutta schemes.

4.1 Runge-Kutta Schemes

$$u_t(x, t) = u_x(x, t) \quad (18)$$

or on a grid we get,

$$\vec{u}_t(t) = D\vec{u}(t) \quad (19)$$

and this has the solution

$$\vec{u} = e^{Dt}\vec{u}_0. \quad (20)$$

We can think of the time advancement scheme in terms of the expansion of the exponential e^{Dt} , i.e.,

$$\vec{u}(\Delta t) = (I + D\Delta t + \dots + \frac{(D\Delta t)^N}{N!} + \frac{(D\Delta t)^{N+1}}{N+1!} e^{D\xi})\vec{u}(0), \quad (21)$$

for some $\xi \in [0, \Delta t]$. Or we can write,

$$\vec{u}(\Delta t) = (I + D\Delta t + \dots + \frac{(D\Delta t)^N}{N!} + TE_{time})\vec{u}_0, \quad (22)$$

In the above expression, D is the exact spatial differentiation operator. An approximate differentiation operator D_{approx} will be such that $D = D_{approx} + D_{error}$. Inserting in the above expression we get,

$$\vec{u}(\Delta t) = (I + (D_{approx} + D_{error})\Delta t + \dots + \frac{(D_{approx} + D_{error}\Delta t)^N}{N!} + TE_{time})\vec{u}_0. \quad (23)$$

One can see from this expression that the leading order sources of error in this Runge-Kutta discretization will be,

$$E_{leadingorder} = D_{error}\Delta t + TE_{time}. \quad (24)$$

For practical reasons, let's replace the temporal truncation error with an upper bound when applied to Fourier modes, $u(0) = e^{ikx}$. We get,

$$MaxTE_{time} = \frac{(\Delta t)^{P_t+1}}{(P_t + 1)!} k^{P_t+1}, \quad (25)$$

where P_t is the order of the temporal discretization. We recall from above that upper bound on the spatial differentiation operator is,

$$D_{errormax} = (\Delta x)^{P_s} \frac{(P_s/2)!^2}{(P_s + 1)!} k^{P_s+1} \quad (26)$$

At each time step we can now see that the error is increased by,

$$\Delta Errmax = D_{errormax} \Delta t + MaxTE_{time} \quad (27)$$

so that after N time steps we arrive at an error of

$$E_{total} = N(\Delta Errmax) = N(D_{errormax} \Delta t + MaxTE_{time}), \quad (28)$$

where N is set such that $T_{final} = N\Delta t$. Now, substituting into the above expression we get,

$$E_{total} = N((\Delta x)^{P_s} \frac{(P_s/2)!^2}{(P_s + 1)!} k^{P_s+1}) \Delta t + \frac{(\Delta t)^{P_t+1}}{(P_t + 1)!} k^{P_t+1}. \quad (29)$$

$$E_{total} = \frac{T_{final}}{\Delta t} ((\Delta x)^{P_s} \frac{(P_s/2)!^2}{(P_s + 1)!} k^{P_s+1}) \Delta t + \frac{(\Delta t)^{P_t+1}}{(P_t + 1)!} k^{P_t+1}. \quad (30)$$

Now divide out the Δt to get,

$$E_{total} = T_{final} ((\Delta x)^{P_s} \frac{(P_s/2)!^2}{(P_s + 1)!} k^{P_s+1} + \frac{(\Delta t)^{P_t}}{(P_t + 1)!} k^{P_t+1}). \quad (31)$$

4.2 The Runge-Kutta Order "Misnomer"

With Runge Kutta methods we should make a short comment with respect to the use of the word order. First of all, a p order Runge Kutta time discretization of,

$$\bar{u}_t = D\bar{u} \quad (32)$$

is equivalent to,

$$RK_p = I + D\Delta t + \dots + \frac{(D\Delta t)^p}{p!} \quad (33)$$

which will have a truncation error of the form,

$$TE_{RK_p}(\Delta t) = \frac{(D\Delta t)^{p+1}}{(p + 1)!}. \quad (34)$$

So, if we construct $u(\Delta t)$ from $u(0)$ then our truncation error is on the order of $(\Delta t)^{p+1}$ which could be called a $p + 1$ order approximation. To illustrate, suppose that we begin with an approximation to $u(\Delta t)$ and we now divide Δt by 2 then the approximation to $u(\Delta t/2)$ will certainly have the truncation error written above as TE_{RK_p} . However, originally we wanted an approximation not at $\Delta t/2$ but at Δt requiring a second time step at $\Delta t/2$. In general, as our Δt is reduced in size by N then we must take N time steps to compensate and this process reduces the order by one.

4.3 Balancing Temporal and Spatial Errors

Typically we call the restriction of the time step as a function of the space step the CFL condition. Typical relationships between the time and the space step are $\Delta t = C_1 \Delta x$ and $\Delta t = C_2 (\Delta x)^2$. The following relationship was found above,

$$E_{total} = T_{final} \left((\Delta x)^{P_s} \frac{(P_s/2)!^2}{(P_s + 1)!} k^{P_s+1} + \frac{(\Delta t)^{P_t-1}}{P_t!} k^{P_t} \right), \quad (35)$$

and if we now incorporate a CFL condition into this expression then we will have a better idea of how to choose numerical orders of accuracy in space and time. Assume that $\Delta t = C(\Delta x)^m$ and substitute into the above expression to get,

$$E_{total} = T_{final} \left((\Delta x)^{P_s} \frac{(P_s/2)!^2}{(P_s + 1)!} k^{P_s+1} + \frac{(C(\Delta x)^m)^{P_t}}{(P_t + 1)!} k^{P_t+1} \right). \quad (36)$$

From this expression we can see that an effective numerical scheme is one where the spatial and temporal errors are balanced. In other words, if one has a spectral scheme in space and a, say, 4th order Runge Kutta scheme in time and a CFL such that $\Delta t \approx \Delta x$ then the spatial errors will be orders of magnitude smaller than the temporal errors and the extra work in space could possibly be for nought.

5 Conclusion

This document is meant as a first in a series which will discuss carefully and precisely numerical errors. A careful understanding of numerical errors is critical for proper DNS or all flows but particularly turbulent flows. It is critical that numerical errors have a smaller affect on the flow than, say, a turbulence model. And, for low order numerical schemes this will often not be the case.

References

- [1] G. Dahlquist and A. Bjorck, (1974) "Numerical Methods", Prentice-Hall.
- [2] I. Daubechies, (1988) "Orthonormal Basis of Compactly Supported Wavelets", Comm. Pure Appl. Math., **41** pp. 909-996.
- [3] P.J. Davis, (1975) "Interpolation and Approximation", Dover.