Numerically Estimating Derivatives During Simulations

Respectfully submitted to Modeling, Simulation and Visualization Methods (MSV'11),
a conference of (WORLDCOMP'11)—the 2011 World Congress
in Computer Science, Computer Engineering, and Applied Computing

Gregory V. Bard
Dept. of Mathematics, Statistics and Computer Science
University of Wisconsin—Stout
Menomonie, WI, 54751-0790
Email: gregory.bard@ieee.org
http://www.math.umd.edu/∼bardg/

***Abstract*—When a function's value is known at several distinct points, there exist "numerical differentiation" formulas which provide estimates for the first derivative, and with various error bounds. In fact, many times the numerical differentiation formula chosen must be complex to avoid intolerable error as shown below. However, in computer simulations, as well as in real-time programming, while several past data points might be known, as well as the present data, the future is not yet known or computed!**

This renders those formulas, at least as classically presented, useless for both real-time computing and for simulations. This research began as an inquiry to discover similar formulas usable when only the past and present, but not the future, are known. The author uncovered an extremely obscure corner of numerical analysis, where solutions to these dilemmas are feasible. An extremely simple algorithm can produce suitable formulas, for any number of data points, not only for the first derivative but also higher-order derivatives. Moreover, the data points can be irregularly spaced, and it is straight-forward to calculate the appropriate error terms, and the impact of noisy data.

The standard presentation of these methods would require *Multivariate Calculus*, one or two semesters of *Real Analysis*, and one semester of *Numerical Analysis*. However, many computer science graduate students have only *Calculus I* and *Calculus II*. This is a novel presentation of this extremely obscure topic, meant not only to publicize it among simluation-oriented computer scientists and those working in real-time computing, but also to be comprehensible to someone who has 2–3 semesters of calculus and nothing more. A rigorous proof of correctness is provided in the appendix, as well as sample code for the algorithm in SAGE—the open source competitor to Maple, Mathematica, Matlab and Magma.

***Index Terms*—Simulations, Numerical Differentiation, Discretization Error.**

I. INTRODUCTION

When one has some data, perhaps about the position of an aircraft, at times one wants to calculate the first derivative. Perhaps the aircraft is being modeled by a simulation, and one has position data at discrete time slices separated by h seconds. Obviously, one can estimate the derivative by

$$\frac{f(t)-f(t-h)}{h} \approx f'(t)$$

but because h is an actual length of time, and not infinitesimal, this produces some error, usually called discretization error. In numerical analysis, this is called the "backward difference" formula. In Section V, we will show that its error is $(h/2)f''(t)+\mathcal{O}\left(h^2 f'''(t)\right)$, which is well known.

Many people working in the overlap of computer science and mathematics know that the "center difference" formula

$$\frac{f(t+h)-f(t-h)}{2h} \approx f'(t)$$

produces slightly less error and the much less well-known "four point" formula is

$$\left(\frac{-1}{3}\right)\left[\frac{f(t+2h)-f(t-2h)}{4h}\right]+\left(\frac{4}{3}\right)\left[\frac{f(t+h)-f(t-h)}{2h}\right] \approx f'(t)$$

which is better still. Nonetheless, both of these numerical differentiation formulas are not useful in simulations, because if $f(t)$ represents the present moment, then $f(t+h)$ and $f(t+2h)$ represent the future, which has not been calculated yet, (or is not yet known.)

Accordingly, many simulation experts go with the "backward difference formula" given above, which is wasteful. Instead, we will show that one can use

$$\frac{1}{h}\left[\frac{1}{4}f(t-4h)-\frac{4}{3}f(t-3h)+3f(t-2h)-4f(t-h)+\frac{25}{12}f(t)\right]$$

which while it look strange, requires no knowledge of the future, and produces

$$f'(t)-\frac{h^4}{5}f^{(5)}(t)+\mathcal{O}\left(h^5 f^{(6)}(t)\right)$$

and since $h \approx 10^{-4}$ or 10^{-3} in many applications, there is virtually no discretization error. We will use this example as the "main example" of the paper, and derive it, as well as derive its error terms. Other sample formulas that the algorithm can produce are given in Figure 1.

Such formulas look miraculous but they have an algorithmic derivation. In this paper, we will explore how to construct such formulas that do not require knowledge of the future, for any number of data points, as well as how they can be adapted to irregular spacings, how the error terms can be simply and exactly derived, and how similar formulas can be derived for higher-order derivatives.

Before we describe the algorithm, we will examine the numerical motivation for finding them in Section II and briefly examine some hypothetical applications in Section III. The algorithm itself is given in Figure 2 and is discussed in Section IV, followed by two warm-up calculations given in Section V. Then the technique for calculating the error (and verifying the correctness of the output) is described in Section VI. Next we discuss the impact of noise in Section VII. We give a survey of the classic expositions on numerical differentiation in Section IX, to demonstrate that these techniques are extremely rare inside the numerical analysis community, and so therefore it is unsurprising that they are essentially unknown outside of it.

A rigorous proof of the correctness of the method, along with the error terms, is given in the appendices, but most readers will not necessarily have the background for what is presented there. Instead, the techniques of Section VI allow the reader to verify the formula outputted by the algorithm for any particular inputs, and an example of this is found in Figure 3. It is hoped that the paper, without appendices, is comprehensible to anyone with a full year of calculus, though this is false for the appendix. The appendices can be found in the full version of the paper on the author's webpage, at the URL given at the top of the front page.

A. Notation

As is customary, the second and third derivatives are written $f''(t)$ and $f'''(t)$ while the fourth and fifth are written $f^{(4)}(t)$ and $f^{(5)}(t)$. Furthermore, $f^{(k)}(t)$ means the kth derivative for any $k \geq 1$. Mathematicians, and this paper, denote the upper-left-hand entry of a matrix A as A_{11}, not A_{00}, as would be required by some programming languages. The basic time slice of a simulation will be denoted h. That is to say, the data points available are assumed to be $f(0)$, $f(h)$, $f(2h)$, $f(3h)$, ...

II. Numerical Motivation

If one is using a simulation to compute something, usually it is because no mathematical function to compute it is known. Otherwise, one would not bother with a simulation, except perhaps for pedagogical purposes. Nonetheless, it is useful to explore some explicit example functions to see the discretization error. Here we take two functions which could come up in the analysis of a physical system, in a first-semester course of *Differential Equations*.

We consider the functions

$$\phi(t) = e^{-4t} \qquad \text{and} \qquad \psi(t) = e^{-4t}(\sin 10t)$$

at $t = 1$, and time slice of 10 msec (i.e. $h = 10^{-2}$.) We compare the performance of the backward-difference formula (denoted BDF) and main example (denoted ME). The data is given in Table I.

While 2.03% error could be perhaps overlooked in the case of $\phi(t)$, the 9.12% error perhaps cannot, in the case of $\psi(t)$. Of course, $h = 0.01$ is a bit large, and more typical values would be $h = 0.001$ or $h = 0.0001$. Yet, that means that the computation would have to run $10\times$ or $100\times$ as long.

TABLE I
A Numerical Example

	phi(t)	psi(t)
Actual	-0.073262555555	-0.113824934424
Predicted by ME	-0.073262515448	-0.113828751659
Absolute Error, ME	0.000000040107	-0.000003817234
Relative Error, ME	-0.000054743602%	0.003353601199%
Predicted by BDF	-0.074747540288	-0.124203517934
Absolute Error, BDF	-0.001484984733	-0.010378583510
Relative Error, BDF	2.026935480971%	9.118022832760%

The greater accuracy can then be realized as a tradeoff against running time. Suppose the tolerable error is some fixed amount. If the h required to achieve that amount is much less with a more complex numerical differentiation formula, than another, surely the ratios of the hs is the factor of the speed-up. However, this is overly simplistic, as a researcher can choose the granularity of the simulation for many reasons, not just the discretization error of some rate that is being calculated.

Also, we will explore in Section VII how making h too small can cause both rounding error and instrumentation error to become large.

III. Potential Applications

There are too many uses of calculating derivatives to even be summarized here. However, in this section we present five hypothetical applications which demonstrate the potential uses of being able to handle irregular spacings, being able to achieve high accuracy, or being able to take higher-order derivatives.

First, in guided missile tests, an altimeter is a simple and off-the-shelf component, but taking the derivative accurately would convert this into a *vertical* velocity measurement. Note, this is not the same as airspeed, which while easily measured with a Pitot tube, and includes the *horizontal* component of velocity. In fact, in this case, second, third and fourth derivatives would be useful, as acceleration, jerk, and jostle are important parameters for any guided missile.

Second, in modeling a biological system of three or more competing components (e.g. viruses, red cells, and white cells) the microscope might be connected to a camera, and either an automated system, or more realistically, a graduate student, would count the number of such visible inside a given grid square. These population counts could not be taken too close together in time, otherwise there would be too many photographs to analyze during the length of the experiment. Nonetheless, it might be useful to discover the rates of growth, to a high level of accuracy. This is particularly important if the rates of growth are actually similar, yet it is pivotal to know which is growing faster than which.

Third, from the purchasing records of medieval colleges (e.g. New College, Oxford, established in 1379), it has become fashionable to reverse engineer the prices of commodities [10] [8]. This enables price data to become known where it

$$\frac{1}{h}\left[\frac{-1}{1}f(t-h)+\frac{1}{1}f(t)\right]=f'(t)+\frac{-1}{2}hf^{(2)}(t)+\mathcal{O}\left(t^2f^{(3)}(t)\right)$$

$$\frac{1}{h}\left[\frac{1}{2}f(t-2h)+\frac{-2}{1}f(t-h)+\frac{3}{2}f(t)\right]=f'(t)+\frac{-1}{3}h^2f^{(3)}(t)+\mathcal{O}\left(t^3f^{(4)}(t)\right)$$

$$\frac{1}{h}\left[\frac{-1}{3}f(t-3h)+\frac{3}{2}f(t-2h)+\frac{-3}{1}f(t-h)+\frac{11}{6}f(t)\right]=f'(t)+\frac{-1}{4}h^3f^{(4)}(t)+\mathcal{O}\left(t^4f^{(5)}(t)\right)$$

$$\frac{1}{h}\left[\frac{1}{4}f(t-4h)+\frac{-4}{3}f(t-3h)+\frac{3}{1}f(t-2h)+\frac{-4}{1}f(t-h)+\frac{25}{12}f(t)\right]=f'(t)+\frac{-1}{5}h^4f^{(5)}(t)+\mathcal{O}\left(t^5f^{(6)}(t)\right)$$

$$\frac{1}{h}\left[\frac{-1}{5}f(t-5h)+\frac{5}{4}f(t-4h)+\frac{-10}{3}f(t-3h)+\frac{5}{1}f(t-2h)+\frac{-5}{1}f(t-h)+\frac{137}{60}f(t)\right]=f'(t)+\frac{-1}{6}h^5f^{(6)}(t)+\mathcal{O}\left(t^6f^{(7)}(t)\right)$$

$$\frac{1}{h}\left[\frac{1}{6}f(t-6h)+\frac{-6}{5}f(t-5h)+\frac{15}{4}f(t-4h)+\frac{-20}{3}f(t-3h)+\frac{15}{2}f(t-2h)+\frac{-6}{1}f(t-h)+\frac{49}{20}f(t)\right]=$$
$$f'(t)+\frac{-1}{7}h^6f^{(7)}(t)+\mathcal{O}\left(t^7f^{(8)}(t)\right)$$

Fig. 1. Numerical Differentiation Formulas that use Past and Present but not Future, as were found by the algorithm in Figure 2.

would otherwise be unavailable. But because such 600-year-old purchasing records have missing "bits", where a record or entry is missing or has been destroyed, techniques are needed which can deal with "lacunae," which are gaps in the data. When the data is sequential, formulas like those given in Figure 1 can be used, but when the data has lacunae, then the algorithm can be called to generate a custom formula that does not involve the missing data point.

Fourth, many financial instruments are traded around the clock, "24-7," but the trading is divided into Monday through Friday periods during normal hours, when the market is open, as well as trading that occurs while the market is closed in the evening and night or weekends. The trading while the market is open results in data that is freely available through web-based tools such as "Yahoo! Finance," whereas the after-hours data is not as easily available in some cases. Let $f(t)$ be the closing price of a financial security on a given day. On a Wednesday, the 5 most-recent data points would represent $f(t)$, $f(t-h)$, $f(t-2h)$, $f(t-5h)$ and $f(t-6h)$, being namely the present day, last Tuesday, last Monday, last Friday, and last Thursday, if h is one day. Thus one needs to be able to take the derivatives when the samples are irregularly spaced.

Fifth, the problem of a rocket taking off is a classic in any *Differential Equations* course. When the instructor includes one feature of realism, such as a rocket meeting with air-resistance, the reduction in mass because of the burnt fuel, or the declining strength of gravity at very high altitudes, then the problem becomes somewhat challenging. But when two or three of these phenomena are included, then the problem is analytically intractable, but can be addressed via a simulation. Such a simulation, at each time slice, would calculate several forces, divide by the instantaneous mass, and then get the acceleration. Then numerical differentiation could be used to calculate the jerk and jostle.

IV. The Algorithm

The algorithm below is more general, as it can find second, third, or arbitrarily higher derivatives. For the kth derivative, at least $k+1$ data points are needed. The spacing of the data points can be highly irregular. The notation used is $f(t+\delta_i h)$, and in a simulation the $\delta_i \leq 0$, because the future is not yet known. For example, the formula presented in Section 2 used the present moment and four previous time slices, as data points. This would be

$$\delta_1=-4,\quad \delta_2=-3,\quad \delta_3=-2,\quad \delta_4=-1,\quad \delta_5=0$$

in this notation. Note, that each δ must be distinct, i.e. no two δs can be equal.

The algorithm, given in Figure 2, is nothing more than a matrix problem of the form $A\vec{c}=\vec{b}$ and in fact, it is a "Vandermonde" matrix [11] named for Alexandre-Théophile Vandermonde. This matrix is shown in Figure 4 of the appendices, and will appear in the proof of correctness of the algorithm. The algorithm, while simple, is probably difficult to understand without reading the proof, but we can verify the validity of its output easily for any particular inputs. In fact, we will do so, in Figure 3, for the "main example" of the paper. For any other inputs, the verification would proceed extremely similarly. Most readers will want to skip the proof.

As stated earlier, only exceptionally pathological functions will fail to be "amenable of type (h,n)", a term defined in Appendix A, and dealt with in Lemma 1. Such functions are very unlikely to arise in applications. This is an obscure technical requirement to remove pathological cases.

A. Numerical Warning on Reducing A:

The bad news is that Vandermonde matrices are known to have extraordinarily high condition number for their size [11], so much so that they are used to "stress test" computer algebra systems before shipment. However, since n is the number of data points used by the formula, and in any practical applied scenario, $n < 11$, this will not be a problem. Computer algebra packages like Maple [1] or SAGE [2] can solve these matrix problems exactly, and not using floating-point notation at all, thus avoiding rounding error entirely. To be precise, the entries of the matrix and both vectors will be rational numbers

Input: A spacing of samples, δ_1, δ_2, δ_3, ..., δ_n, all distinct, and some integer k such that $1 \leq k < n$.
Output: A vector $\vec{c}$ such that

$$\frac{1}{h^k}\sum_{j=1}^{j=n} c_j f(t+\delta_j h) = f^{(k)}(t) + \mathcal{O}\left(h^{n-k} f^{(n)}(t)\right)$$

for any function $f(t)$, that is "amenable of type (h, n)."

1) Define a $n \times n$ matrix A such that $A_{ij} = (\delta_j)^{i-1}$.
2) Define an n-dimensional vector $\vec{b}$, and let $\vec{b} = \vec{0}$, except that $b_{k+1} = k!$.
3) Via Gaussian Elimination or some other means, find $\vec{c}$ such that $A\vec{c} = \vec{b}$.
   Note: Such a $\vec{c}$ always exists and is unique.
4) Return $\vec{c}$.

Fig. 2. The algorithm that produced the formulas in Figure 1.

represented exactly, as the ratio of two explicit integers which are themselves stored as binary strings. For even medium-sized problems, this is normally infeasible, but these A will not have more than a hundred entries or so, and therefore there is no difficulty in using this exact representation.

V. Two Simple Examples

We start with Taylor's Theorem from calculus, an important infinite sum:

$$f(t+h) = f(t) + \sum_{i=1}^{i=\infty} \frac{h^i}{i!} f^{(i)}(t)$$

which can be truncated early for $h \ll 1$, to produce a finite sum, using big-Oh notation

$$f(t+h) = f(t) + \left(\sum_{i=1}^{i=n} \frac{h^i}{i!} f^{(i)}(t)\right) + \mathcal{O}\left(h^{n+1} f^{(n+1)}(t)\right)$$

and note that this is the same big-Oh used when explaining to students that sorting is $\mathcal{O}(n \log n)$ time. It is justified because every term after the nth will be a multiple of h^{n+1}, and since $h \ll 1$ then the h^{n+1} term will dominate the sum. We will clarify precisely when this use of big-Oh notation is correct in the formal proof given in Appendix A. For now, we point out that the big-Oh, instead of being as the data-input length goes to infinity, represents the granularity of the calculation (here h) going to zero.

For the backward-difference we select $n = 2$ and then have

$$f(t+h) = f(t) + hf'(t) + \frac{h^2}{2} f''(t) + \mathcal{O}\left(h^3 f'''(t)\right)$$

and so

$$\frac{f(t+h) - f(t)}{h} = f'(t) + \frac{h}{2} f''(t) + \mathcal{O}\left(h^2 f'''(t)\right)$$

therefore the error is $(h/2)f''(t) + \mathcal{O}\left(h^2 f'''(t)\right)$.

Meanwhile, for the "main example," we offer two roads to proof. The main example is, of course, a special case of the more general theorem. However, a direct proof is provided in Figure 3, for two reasons. First, the rigorous proof of the theorem is relatively arcane and might be easier to understand after looking over the more calculation-oriented proof in Figure 3. Second, if an undergraduate must read this paper, note that the formal proof requires some facts learnt during *Real Analysis* and *Linear Algebra*, where as the simpler proof in the figure requires only Taylor's Theorem, which is usually taught in *Calculus II* or sometimes *Calculus III* or even *Calculus I*.

A. Note on the Error Terms:

Typically, the error term of Taylor's Theorem, for an nth-degree Taylor polynomial approximating $f(t+h)$ by expansion around $f(t)$ is stated via

$$f(t) + \sum_{i=1}^{i=n} \frac{h^i}{i!} f^{(i)}(t) = f(t+h) - \frac{h^n}{n!} f^{(n+1)}(\xi)$$

where ξ is some number between t and $t+h$.

However, this is non-descriptive for us for several reasons. First, the computer science audience is already extremely familiar with Big-Oh notation. Second, we wish to model derivatives of $f(t)$ and so our error bounds really should be in terms of those derivatives, and not f itself, otherwise one is comparing apples and oranges. Third, a happy and very unexpected coincidence is that the proof works out to be much simpler in Big-Oh notation. Naturally, this is not the first use of Big-Oh notation in real analysis, as Landau is known to have used Big-Oh notation between the world wars [12].

VI. Exploring the Discretization Error

It turns out that the error can be explored very precisely, independent of the notions of "amenability" which make the proof of the main theorem a bit complicated. All that is needed is to extend the Taylor polynomial a few extra terms, and calculate a particular double summation. We begin again, with Taylor's Theorem, but now with extra terms:

$$f(t+\delta_1 h) = f(t) + \sum_{i=1}^{i=n+4} \frac{\delta_1^i h^i}{i!} f^{(i)}(t) + \mathcal{O}\left(h^{n+5} f^{(n+5)}(t)\right)$$

and now sum this over all n data points

$$\sum_{j=1}^{j=n} c_j f(t+\delta_j h) = \sum_{j=1}^{j=n} c_j f(t) + \sum_{j=1}^{j=n} \sum_{i=1}^{i=n+4} \frac{c_j \delta_j^i h^i}{i!} f^{(i)}(t) + \mathcal{O}\left(h^{n+5} f^{(n+5)}(t)\right) \quad (1)$$

and from the double summation in Equation (1) you can see that the coefficient of $h^i f^{(i)}(t)$, to be denoted E_i, is given by

$$E_i = \sum_{j=1}^{j=n} \frac{c_j \delta_j^i}{i!}$$

We start with Taylor's Theorem, restricted to seven terms, which is

$$f(t+h) = f(t) + hf'(t) + \frac{h^2}{2}f''(t) + \frac{h^3}{3!}f'''(t) + \frac{h^4}{4!}f^{(4)}(t) + \frac{h^5}{5!}f^{(5)}(t) + \frac{h^6}{6!}f^{(6)}(t) + \mathcal{O}\left(h^7 f^{(7)}(t)\right)$$

and apply this to each of $a \in \{0, -h, -2h, -3h, -4h\}$ and thus obtain

$$\begin{aligned}
f(t-4h) &= f(t) - \frac{4}{1!}hf'(t) + \frac{16}{2!}h^2f''(t) - \frac{64}{3!}h^3f'''(t) + \frac{256}{4!}h^4f^{(4)}(t) - \frac{1024}{5!}h^5f^{(5)}(t) + \frac{4096}{6!}h^6f^{(6)}(t) + \mathcal{O}\left(h^7f^{(7)}(t)\right) \\
f(t-3h) &= f(t) - \frac{3}{1!}hf'(t) + \frac{9}{2!}h^2f''(t) - \frac{27}{3!}h^3f'''(t) + \frac{81}{4!}h^4f^{(4)}(t) - \frac{243}{5!}h^5f^{(5)}(t) + \frac{729}{6!}h^6f^{(6)}(t) + \mathcal{O}\left(h^7f^{(7)}(t)\right) \\
f(t-2h) &= f(t) - \frac{2}{1!}hf'(t) + \frac{4}{2!}h^2f''(t) - \frac{8}{3!}h^3f'''(t) + \frac{16}{4!}h^4f^{(4)}(t) - \frac{32}{5!}h^5f^{(5)}(t) + \frac{64}{6!}h^6f^{(6)}(t) + \mathcal{O}\left(h^7f^{(7)}(t)\right) \\
f(t-1h) &= f(t) - \frac{1}{1!}hf'(t) + \frac{1}{2!}h^2f''(t) - \frac{1}{3!}h^3f'''(t) + \frac{1}{4!}h^4f^{(4)}(t) - \frac{1}{5!}h^5f^{(5)}(t) + \frac{1}{6!}h^6f^{(6)}(t) + \mathcal{O}\left(h^7f^{(7)}(t)\right) \\
f(t-0h) &= f(t)
\end{aligned}$$

and therefore we can, with much effort and patience, substitute those formulae into

$$\begin{aligned}
&\frac{1}{4}f(t-4h) - \frac{4}{3}f(t-3h) + 3f(t-2h) - 4f(t-h) + \frac{25}{12}f(t) \\
&= \left[\left(\frac{1}{4}\right)(1) - \left(\frac{4}{3}\right)(1) + (3)(1) - (4)(1) + \left(\frac{25}{12}\right)(1)\right] f(t) \\
&\quad + \left[\left(\frac{1}{4}\right)\left(\frac{-4}{1!}\right) - \left(\frac{4}{3}\right)\left(\frac{-3}{1!}\right) + (3)\left(\frac{-2}{1!}\right) - (4)\left(\frac{-1}{1!}\right) + \left(\frac{25}{12}\right)\left(\frac{0}{1!}\right)\right] hf'(t) \\
&\quad + \left[\left(\frac{1}{4}\right)\left(\frac{16}{2!}\right) - \left(\frac{4}{3}\right)\left(\frac{9}{2!}\right) + (3)\left(\frac{4}{2!}\right) - (4)\left(\frac{1}{2!}\right) + \left(\frac{25}{12}\right)\left(\frac{0}{2!}\right)\right] h^2f''(t) \\
&\quad + \left[\left(\frac{1}{4}\right)\left(\frac{-64}{3!}\right) - \left(\frac{4}{3}\right)\left(\frac{-27}{3!}\right) + (3)\left(\frac{-8}{3!}\right) - (4)\left(\frac{-1}{3!}\right) + \left(\frac{25}{12}\right)\left(\frac{0}{3!}\right)\right] h^3f'''(t) \\
&\quad + \left[\left(\frac{1}{4}\right)\left(\frac{256}{4!}\right) - \left(\frac{4}{3}\right)\left(\frac{81}{4!}\right) + (3)\left(\frac{16}{4!}\right) - (4)\left(\frac{1}{4!}\right) + \left(\frac{25}{12}\right)\left(\frac{0}{4!}\right)\right] h^4f^{(4)}(t) \\
&\quad + \left[\left(\frac{1}{4}\right)\left(\frac{-1024}{5!}\right) - \left(\frac{4}{3}\right)\left(\frac{-243}{5!}\right) + (3)\left(\frac{-32}{5!}\right) - (4)\left(\frac{-1}{5!}\right) + \left(\frac{25}{12}\right)\left(\frac{0}{5!}\right)\right] h^5f^{(5)}(t) \\
&\quad + \left[\left(\frac{1}{4}\right)\left(\frac{4096}{6!}\right) - \left(\frac{4}{3}\right)\left(\frac{729}{6!}\right) + (3)\left(\frac{64}{6!}\right) - (4)\left(\frac{1}{6!}\right) + \left(\frac{25}{12}\right)\left(\frac{0}{6!}\right)\right] h^6f^{(6)}(t) + \mathcal{O}\left(h^7f^{(7)}(t)\right) \\
&= 0f(t) + 1hf'(t) + 0h^2f''(t) + 0h^3f'''(t) + 0h^4f^{(4)}(t) - \frac{1}{5}h^5f^{(5)}(t) + \frac{1}{3}h^6f^{(6)}(t) + \mathcal{O}\left(h^7f^{(7)}(t)\right) \\
&= hf'(t) - \frac{h^5}{5}f^{(5)}(t) + \mathcal{O}\left(h^6f^{(6)}(t)\right)
\end{aligned}$$

allowing us to finally conclude (by dividing both sides by h) that

$$\frac{1}{h}\left[\frac{1}{4}f(t-4h) - \frac{4}{3}f(t-3h) + 3f(t-2h) - 4f(t-h) + \frac{25}{12}f(t)\right] = f'(t) - \frac{h^4}{5}f^{(5)}(t) + \mathcal{O}\left(h^5f^{(6)}(t)\right)$$

and while this was not the shortest calculation in human history, since $t \approx 10^{-3}$ or 10^{-4} in most cases, then h^4 makes the error *completely* negligible. That is, of course, provided that the fifth derivative of $f(t)$ is not huge, which is why $f(t) = \sin\left(10^6 t\right)$ cannot be used here. This will be sorted out in the appendices via Lemma 1.

Fig. 3. An Elementary Approach to Proving the "main example," suitable for an undergraduate who has taken one year of calculus.

which is a sum that can be computed, either with a pencil or a computer algebra package. In fact, if one adopts the syntax convention that the 0th derivative of $f(t)$ is $f(t)$ itself, and that 0!=1, then this remains true for the first sum to the right of the equal sign as well, in Equation (1).

Using this, we can compute the following for the "main example" of the paper:

$$\begin{aligned}\frac{1}{h}&\left[\frac{1}{4}f(t-4h)-\frac{4}{3}f(t-3h)+3f(t-2h)-4f(t-h)+\frac{25}{12}f(t)\right]\\ &= 0f(t)+1tf'(t)+0h^2f''(t)+0h^3f'''(t)+0h^4f^{(4)}(t)\\ &\quad -\frac{1}{5}h^5f^{(5)}(t)+\frac{1}{3}h^6f^{(6)}(t)-\frac{13}{42}h^7f^{(7)}(t)+\frac{5}{24}h^8f^{(8)}(t)\\ &\quad -\frac{9}{80}h^9f^{(9)}(t)+\mathcal{O}\left(h^{10}f^{(10)}(t)\right)\end{aligned}$$

and the SAGE code that did this is given in the next section. Keep in mind that $h = 10^{-4}$ is a common value, so the h^6th and higher terms are *completely* negligible.

VII. Exploring the Impact of Noise

The above work is all performed under the assumption that the functions are evaluated without error (e.g. a computer simulation.) Errors in function evaluation can be noise in the data (due to instrumentation issues), or rounding error. This is distinct from the discretization error that the paper principally addresses. However, because both rounding error and instrumentation error are inescapable to some degree, we will now explore their impact on the accuracy of these methods.

Suppose a particular evaluation of f is off by a relative error of ϵ. Then we can replace $f(t)$ with $(1+\epsilon)f(t)$. To be general, we can select any of the $f(t+h\delta_i)$. Then we have

$$\frac{1}{h^k}\left[\sum_{i=1}^{i=n} c_i f\left(t+h\delta_i\right)\right]$$

...becomes...

$$\begin{aligned}&\frac{1}{h^k}\left[c_j(1+\epsilon)f(t+h\delta_j)+\sum_{i=1,i\neq j}^{i=n} c_i f\left(t+h\delta_i\right)\right]\\ &= \frac{1}{h^k}\left[c_j f(t+h\delta_j)+c_j\epsilon f(t+h\delta_j)+\sum_{i=1,i\neq j}^{i=n} c_i f\left(t+h\delta_i\right)\right]\\ &= \frac{c_j\epsilon}{h^k}f(t+h\delta_j)+\frac{1}{h^k}\sum_{i=1}^{i=n} c_i f\left(t+h\delta_i\right)\end{aligned}$$

And so the absolute induced error is

$$\text{absolute error} = \frac{c_j\epsilon f(t+h\delta_j)}{h^k}$$

and under the quite reasonable assumption that all the f's are of approximately the same magnitude, then the relative error is

$$\text{relative error} = \left(\frac{\epsilon}{h^k}\right)\left(\frac{c_j}{\sum_{i=1}^{i=n} c_i}\right)$$

As you can see from that formula, the dominant effect will be that the relative instrumentation or rounding error is magnified by a multiplicative factor proportional to h^{-k}. Since $k > 1$, then using too small h will make any errors of these kinds catastrophic.

In Section V, we calculated that the error for the backward-difference formula was proportional to $(h/2)f''(t)$, and for the "main example" it was proportional to $(h^4/5)f^{(5)}(t)$. This means a user of the main example who uses $h = 0.001$ can, under the assumption that $f''(t) \approx f^{(5)}(t)$ have the same discretization error as a user of the backward difference formula and $h = 4 \times 10^{-13}$. Likewise, for $h = 0.01$ in the main example, it is equivalent to using $h = 4 \times 10^{-9}$.

But, the consequences of this would be that instead of an instrumentation error being multiplied by a factor of 1000, it would be multiplied by trillions or billions.

Rounding error in computer systems using only `double` floating-point numbers, to say nothing of `long double`, are on the order of 2^{-53} or $10^{-15.9545\cdots}$ and so a magnification of $1000\times$ is acceptable, while a magnification of several trillion is not.

On the other hand, most scientific instruments are not capable of more than 4 significant digits, and so very large h, e.g. $h \approx 0.1$ should be chosen, to keep instrumentation error low.

VIII. Sample SAGE Code

The following SAGE code will execute the algorithm given in Figure 2 as well as the error modeling as described in Section VI. As you can see, the former is just a matrix computation, though A has a very special form, and the latter is just a double summation. The proof of correctness is in the appendices. This SAGE code was used to produce the formulas found in Figure 1.

More information about SAGE can be found at [2].

```
n=5
k=1
d=[-4,-3,-2, -1,  0];
debug=false

# don't modify what comes below this point.

A=matrix(QQ, n,n);
b=matrix(QQ, n,1);
for i in range(0, n):
    b[i]=0
    for j in range(0, n):
        A[i, j] = d[j]^(i)
b[k]=factorial(k)
c=A.inverse()*b

if (debug==true):
    print "The Matrix A:"
    print A
    print "The Vector b:"
    print b
    print "The Vector c:"
    print c

for i in range(0, n):
    print "f(t + (",
    print d[i],
    print ")h) is multiplied by ",
    print c[i]
```

```
print "To yield:"

for i in range(0,n+5):
    answer=0;
    for j in range(0,n):
        answer += c[j-1] * (d[j-1]^i) / factorial(i)
    if (i>0):
        print "For t^%d times %dth derivative:" % \
                                  ((i-k),i),
    else:
        print "For t^%d times the function: " % \
                                 (i-k),
    print answer
```

Note that the `c[j-1]` has a $j-1$ because the computer language "Python," upon which SAGE is built, numbers its array entries with the first component being 0 and not 1, likewise `d[j-1]`.

IX. Prior and Related Work

In order to demonstrate the obscurity of these methods, as well as to give the reader further reading, a survey now follows which covers the most famous textbooks that would be used in a first-year graduate course in *Numerical Analysis* or a senior-year course for undergraduates at a highly-ranked university.

In [14, Ch. 5.7], usually the surest refuge of interesting and rarely known algorithms, there is a discussion of numerical derivatives, but there is no mention whatsoever of formulas which avoid data points in the future, or even derivatives taken with more than 3 evaluations of $f(t)$, nor methods for second derivatives.

On the other hand, in [3, Ch. 5.4] the general technique of this paper is almost, but not quite, discussed under the title "method of undetermined coefficients," a name which numerous mathematical techniques share. That book does not cover the case of irregularly spacings, and only considers first and second derivatives. It makes no allowance for formulas which avoid data points in the future. They include formulas with 5 evaluations of $f(t)$, but none with more than 5 evaluations. The solving for the coefficients is not treated as a matrix problem, and so it is unsurprising that they miss the connection with Vandermonde matrices.

The text [9, Ch. 4.9] does provide formulas for numerical differentiation, including formulas with 5 evaluations to $f(t)$ but none with more. They do consider formulas which only refer to data points on one side of t—the only book the author found to have done so. They only consider the first and second derivative, and sadly, the formulas are not derived at all, to say nothing of rigor. This is perhaps understandable, as this text was written to be the "soft core" version of the following book, by the same authors.

Meanwhile, [6, Ch. 4.1] is the "hard core" version of the book in the previous paragraph, by the same authors. The formulas are derived, but much differently (using Lagrange Polynomials), and the authors restrict to the case of f' and f'' only. They do consider formulas with 5 points, but none larger, and they say nothing about formulas using only past and present but not future points.

While all the books previously mentioned ignore derivatives above the second, the text [7, Ch. 4.3] does go higher, and does do non-symmetric spacings of the data. The derivation there is from the Taylor Series, but that book does not consider formulas using only past and present but not future points.

Last but not least, the classic [13, Ch. 7] has a section on numerical differentiation, but no mention of formulas with irregular spacing, nor those that do not require knowledge of the future. It also does not consider those formulas that utilize more than 3 points.

The approach to handling noisy data was taken from [4].

Acknowledgments

The author would like to thank the National Science Foundation, Division of Mathematical Sciences, for Grant Number # `DMS-0821725`, awarded to Prof. William Stein and the SAGE community, as SAGE played a pivotal role in this research. This includes certain computational resources at the University of Washington in Seattle which Prof. Stein permitted the author to use, including for simulations.

The author would also like to thank Prof. Peter Wolfe, of the University of Maryland Department of Mathematics, who taught him *Numerical Analysis*, and whose perspective on the subject in general has guided this researcher's work on several topics.

Appendix

The appendices do not fit within the 7-page requirement of the conference, but can be found in the full version of the paper, available on the author's webpage. The URL is found on the front page of this paper.

References

[1] Maple. Software Package. Available at http://www.maplesoft.com/
[2] SAGE. Software Package. Available at http://www.sagemath.org/
[3] Kendall Atkinson and Weimin Han. "Ch. 5.4.2: Numerical Differentiation via The Method of Undetermined Coefficients." *Elementary Numerical Analysis*, 3rd ed. John Wiley and Sons. 2004.
[4] as above, but "Ch. 5.4.3: Effects of Error in Function Values."
[5] R. Creighton Buck. "Ch 3.5: Taylor's Theorem." *Advanced Calculus.* McGraw-Hill Book Company. 1978.
[6] Richard Burden and J. Douglas Faires. "Ch 4.1: Numerical Differentiation." *Numerical Analysis*. 8th ed. Thomson, Brooks and Cole. 2005.
[7] Ward Cheney and David Kincaid. "Ch. 4.3: Estimating Derivatives and Richardson Extrapolation." *Numerical Mathematics and Computing*. 6th ed. Thomson, Brooks and Cole. 2008.
[8] Christopher Dyer. *Standards of Living in the Later Middle Ages*. Cambridge University Press, 1989.
[9] J. Douglas Faires and Richard Burden. "Ch. 4.9: Numerical Differentiation." *Numerical Methods*. 3rd ed. Thomson, Brooks and Cole. 2003.
[10] Elizabeth Gemmill and Nicholas Mayhew. *Changing Values in Medieval Scotland: A Study of Prices, Money, and Weights and Measures*. Cambridge University Press. 1995.
[11] Gene Golub and Charles van Loan. "Ch. 4.6: Vandermonde Systems and the FFT." *Matrix Computations.* 3rd ed. Johns Hopkins University Press. 1996.
[12] Edmund Landau. *Foundations of Analysis.* 1930.
[13] Francis Scheid. "Ch. 7. Numerical Differentiation." *2000 Solved Problems in Numerical Analysis (Schaum's Solved Problem Series)*. McGraw-Hill. 1990.
[14] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. "Ch 5.7: Numerical Derivatives." *Numerical Recipes in C*. 2nd ed. Cambridge University Press. 1992.

APPENDIX

We will now give the rigorous proof that the algorithm is correct. Most readers will want to skip this, as it is not necessary to understand why the algorithm works in general, because the reader can verify its output for any particular input. In fact, this was done in Figure 3.

A. The Derivation of the Algorithm

Suppose one has the data points $f(t+\delta_1 h)$, $f(t+\delta_2 h)$, $f(t+\delta_3 h)$, …, $f(t+\delta_n h)$, and one wants to find the kth derivative of f at t. We require $k<n$, and that f be n-times differentiable, which is to say that $f^{(n)}(t)$ exists but is not necessarily continuous. The objective is to find c_1, c_2, c_3, …, c_n, such that

$$\frac{c_1}{h^k}f(t+\delta_1 h)+\frac{c_2}{h^k}f(t+\delta_2 h)+\cdots+\frac{c_n}{h^k}f(t+\delta_n h)\approx f^{(k)}(t)$$

with error terms small and precisely calculated.

We start with Taylor's Formula

$$f(t+\delta_1 h)=f(t)+\sum_{i=1}^{i=\infty}\frac{\delta_1^i h^i}{i!}f^{(i)}(t)$$

which we simplify slightly by adopting the (extremely standard) syntax conventions that $0!=1$ and $f^{(0)}(t)=f(t)$. That is to say, the 0th derivative of a function is the function itself. We have

$$f(t+\delta_1 h)=\sum_{i=0}^{i=\infty}\frac{\delta_1^i}{i!}h^i f^{(i)}(t)$$

and therefore

$$\sum_{j=1}^{j=n}c_j f(t+\delta_j h)=\mathcal{S}=\sum_{j=1}^{j=n}\left[\sum_{i=0}^{i=\infty}c_j\frac{\delta_j^i}{i!}h^i f^{(i)}(t)\right]$$

where $\mathcal{S}$ is just an abbreviation, to save space typesetting the equations. We can split the infinite sum into two parts getting

$$\mathcal{S}=\sum_{j=1}^{j=n}\left[\sum_{i=0}^{i=n-1}c_j\frac{\delta_j^i}{i!}h^i f^{(i)}(t)+\sum_{i=n}^{i=\infty}c_j\frac{\delta_j^i}{i!}h^i f^{(i)}(t)\right]$$

and now intuition might lead one to believe that the right-hand-most sum can be discarded if we assume that $t<1$ and that the higher-order derivatives of $f(t)$ are sufficiently small. Informally, each of the terms in the right-hand series is a multiple of t^n, and if $t\ll 1$ then each successive term is much smaller than all the terms before it. Thus, intuition would lead one to believe that the entire sum is $\mathcal{O}(t^n)$. In order to do this formally, we introduce a new definition:

Definition 1: If there is a real number $0\leq v<1$ such that for all $r>n$

$$\left|h^r f^{(r)}(t)\right|<\left|v^{r-n}h^n f^{(n)}(t)\right|$$

then we say that "$f(t)$ is amenable of type (h,n)."

By Lemma 1, proven in Appendix D, for any $f(t)$ that is amenable of type (h,n) one has

$$\sum_{i=n}^{i=\infty}h^r f^{(r)}(t)=\mathcal{O}\left(h^n f^{(n)}(t)\right)$$

with the big-Oh being taken as h goes to zero, and $f^{(n)}(t)$ varies in any way. This is a very minor technicality and most readers will want to skip the proof of the lemma, but we provide a simple example in Appendix E. In any case, if $f(t)$ is amenable of type (h,n), then the first term of that right-hand-most series will dominate, and we have

$$\mathcal{S}=\sum_{j=1}^{j=n}\left[\sum_{i=0}^{i=n-1}c_j\frac{\delta_j^i}{i!}h^i f^{(i)}(t)+\mathcal{O}\left(h^n f^{(n)}(t)\right)\right]$$

From this point, we make two observations. First, that the big-Oh term has no i or j in it, and so it can be evicted from both series (i.e. pulled across both large sigmas as a constant).[1] Second, all summations are now finite sums, and so we can reorder the terms at will—an action which is not possible with infinite sums without use of theorems that distinguish between uniform and non-uniform convergence. But since these terms are finite, we merely interchange the two summations on the right of the equal sign to get

$$\mathcal{S}=\mathcal{O}\left(h^n f^{(n)}(t)\right)+\sum_{i=0}^{i=n-1}\left[\sum_{j=1}^{j=n}c_j\frac{\delta_j^i}{i!}h^i f^{(i)}(t)\right]$$

and finally, since $h^i f^{(i)}(t)/(i!)$ does not contain the symbol j, we evict it out of the innermost sum to obtain

$$\mathcal{S}=\mathcal{O}\left(h^n f^{(n)}(t)\right)+\sum_{i=0}^{i=n-1}\left[\frac{1}{i!}h^i f^{(i)}(t)\sum_{j=1}^{j=n}c_j\delta_j^i\right]$$

Surely, then, if one were to find such c_1, c_2, c_3, …, c_n, such that for all $i\neq k$, the right-hand-most sum were 0, and such that for $i=k$, the right-hand-most sum were $i!$, then

$$\sum_{j=1}^{j=n}c_j f(t+\delta_j h)=\mathcal{S}=\mathcal{O}\left(h^n f^{(n)}(t)\right)+h^k f^{(k)}(t)$$

and therefore

$$\frac{1}{h^k}\sum_{j=1}^{j=n}c_j f(t+\delta_j h)=f^{(k)}(t)+\mathcal{O}\left(h^{n-k}f^{(n)}(t)\right)$$

as is desired. All that is needed is to find those c_1, c_2, c_3, …, c_n. We therefore require

$$\sum_{j=1}^{j=n}c_j\delta_j^i=0$$

for all $i\in\{0,1,2,\ldots,k-2,k-1,k+1,k+2,\ldots,n-1\}$ and also

$$\sum_{j=1}^{j=n}c_j\delta_j^k=k!$$

[1] One is tempted to write $\mathcal{O}\left(n^2h^n f^{(n)}(t)\right)$ because there are n^2 appearances of this term. However, for any particular usage of this technique, n is fixed, and so n^2 is merely a constant. In any case, we derive better error terms in Section VI. Nonetheless, the n^2 magnification of the error should give the user a moment of pause before exchanging a 5-data-point formula with an 11-data-point formula.

$$\begin{bmatrix} \delta_1^0 & \delta_2^0 & \delta_3^0 & \cdots & \delta_{k-1}^0 & \delta_k^0 & \delta_{k+1}^0 & \cdots & \delta_{n-1}^0 & \delta_n^0 \\ \delta_1^1 & \delta_2^1 & \delta_3^1 & \cdots & \delta_{k-1}^1 & \delta_k^1 & \delta_{k+1}^1 & \cdots & \delta_{n-1}^1 & \delta_n^1 \\ \delta_1^2 & \delta_2^2 & \delta_3^2 & \cdots & \delta_{k-1}^2 & \delta_k^2 & \delta_{k+1}^2 & \cdots & \delta_{n-1}^2 & \delta_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \delta_1^{k-1} & \delta_2^{k-1} & \delta_3^{k-1} & \cdots & \delta_{k-1}^{k-1} & \delta_k^{k-1} & \delta_{k+1}^{k-1} & \cdots & \delta_{n-1}^{k-1} & \delta_n^{k-1} \\ \delta_1^k & \delta_2^k & \delta_3^k & \cdots & \delta_{k-1}^k & \delta_k^k & \delta_{k+1}^k & \cdots & \delta_{n-1}^k & \delta_n^k \\ \delta_1^{k+1} & \delta_2^{k+1} & \delta_3^{k+1} & \cdots & \delta_{k-1}^{k+1} & \delta_k^{k+1} & \delta_{k+1}^{k+1} & \cdots & \delta_{n-1}^{k+1} & \delta_n^{k+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \delta_1^{n-2} & \delta_2^{n-2} & \delta_3^{n-2} & \cdots & \delta_{k-1}^{n-2} & \delta_k^{n-2} & \delta_{k+1}^{n-2} & \cdots & \delta_{n-2}^{n-1} & \delta_n^{n-2} \\ \delta_1^{n-1} & \delta_2^{n-1} & \delta_3^{n-1} & \cdots & \delta_{k-1}^{n-1} & \delta_k^{n-1} & \delta_{k+1}^{n-1} & \cdots & \delta_{n-1}^{n-1} & \delta_n^{n-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_k \\ c_{k+1} \\ c_k \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ k! \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Fig. 4. The matrix problem mentioned in the algorithm, and the formal proof.

which we can write much more simply as a matrix problem, given in Figure 4, and ask some computer-algebra software package to solve it for us.

This matrix has several interesting properties. First, the ijth entry is δ_j^{i-1}. Second, this is the transpose of a Vandermonde matrix [11]. The good news is that a Vandermonde matrix always has an inverse (provided that $\delta_i \neq \delta_j$ when $i \neq j$), and so this system of equations always has exactly one solution. This is why we required all the δs to be distinct. Since the matrix is invertible, we are assured that the c_1, c_2, c_3, …, c_n will always exist and be unique.

B. Conclusion:

We have now proved

Theorem 1: Let δ_1, δ_2, δ_3, …, δ_n, be distinct and real, and let $1 \leq k < n$. Furthermore, let $f(t)$ be an n-times differentiable function of the real line that is amenable of type (h, n), as defined in Lemma 1. Define A to be an $n \times n$ matrix such that $A_{ij} = \delta_j^{i-1}$, and $\vec{b}$ to be an n-dimensional vector such that all entries are 0, except the $(k+1)$th, which equals $k!$. Then

$$\frac{1}{h^k} \sum_{j=1}^{j=n} c_j f(t + \delta_j h) = f^{(k)}(t) + \mathcal{O}\left(h^{n-k} f^{(n)}(t)\right)$$

where $\vec{c}$ is the unique vector such that $A\vec{c} = \vec{b}$.

C. Two Technical Requirements

There were two technical requirements in the theorem. First, is that $f(t)$ be n-times differentiable. This means that $f^{(k)}(t)$ exists and is defined for $1 \leq k \leq n$. That also implies that all of the these except the nth derivative are continuous. In basically any application, the function would be smooth, which means the kth derivative would exist and be continuous for all $k > 1$. The only terms of the Taylor series that we did not throw away during the proof were those up to and including the $(n-1)$th derivative. A Taylor approximation to that many terms requires f to be n-times differentiable [5]. The author would like to emphasize that this unimportant for applications.

The second technical detail is the proof of the lemma.

D. The Proof of Lemma 1

We will now prove the following lemma.

Lemma 1: If there is a real number $0 \leq v < 1$ such that for all $r > n$

$$\left| h^r f^{(r)}(t) \right| < \left| v^{r-n} h^n f^{(n)}(t) \right|$$

then we say that $f(t)$ is amenable of type (h, n). For any $f(t)$ that is amenable of type (h, n), it is the case that

$$\sum_{i=n}^{i=\infty} h^i f^{(i)}(t) = \mathcal{O}\left(h^n f^{(n)}(t)\right)$$

with the big-Oh being taken as h goes to zero, and $f^{(n)}(t)$ varies in any way.

Proof: Surely if $v = 0$ then the terms with the $(n+1)$th and higher derivatives are all zero, and the result is obvious. We now can dispose of the $0 < v < 1$ as follows. We start with the given

$$\left| h^r f^{(r)}(t) \right| < \left| v^{r-n} h^n f^{(n)}(t) \right|$$

so then the summations will maintain that relationship

$$\sum_{i=n}^{i=\infty} \left| h^i f^{(i)}(t) \right| < \sum_{i=n}^{i=\infty} \left| v^{i-n} h^n f^{(n)}(t) \right|$$

and because v is positive, we can remove it from the absolute value signs

$$\sum_{i=n}^{i=\infty} \left| h^i f^{(i)}(t) \right| < \sum_{i=n}^{i=\infty} v^{i-n} \left| h^n f^{(n)}(t) \right|$$

and since the right-hand absolute value term has no i in it, that can migrate across the right-hand sigma, to obtain

$$\sum_{i=n}^{i=\infty} \left| h^i f^{(i)}(t) \right| < \left| h^n f^{(n)}(t) \right| \sum_{i=n}^{i=\infty} v^{i-n}$$

but the right-most sum is now a simple geometric series (which is why we required $0 < v < 1$) giving us

$$\sum_{i=n}^{i=\infty} \left| h^i f^{(i)}(t) \right| < \left| h^n f^{(n)}(t) \right| \frac{1}{1-v}$$

we could divide through by $n!$ to obtain

$$\sum_{i=n}^{i=\infty} \left| \frac{h^i}{n!} f^{(i)}(t) \right| < \left| \frac{h^n}{n!} f^{(n)}(t) \right| \frac{1}{1-v}$$

and we are almost there.

The only flaw is that h^i has $n!$ under its denominator, instead of $i!$. But since $i \geq n$ in that left-hand sum, that implies $i! \geq n!$ and so

$$\frac{1}{i!} \leq \frac{1}{n!} \text{ thus } \sum_{i=n}^{i=\infty} \left| \frac{h^i}{i!} f^{(i)}(t) \right| \leq \sum_{i=n}^{i=\infty} \left| \frac{h^i}{n!} f^{(i)}(t) \right|$$

Then by the triangle inequality

$$\sum_{i=n}^{i=\infty} \left| \frac{h^i}{i!} f^{(i)}(t) \right| < \left| \frac{h^n}{n!} f^{(n)}(t) \right| \frac{1}{1-v}$$

and since v is a constant, so is $1/(1-v)$ a constant. Thus at long last

$$\sum_{i=n}^{i=\infty} \left| \frac{h^i}{i!} f^{(i)}(t) \right| = \mathcal{O}\left(h^n f^{(n)}(t) \right)$$

and since $\sum |y_i| \geq \sum y_i$ for any sequence y_i we can safely remove the absolute values and then we have

$$\sum_{i=n}^{i=\infty} \frac{h^i}{i!} f^{(i)}(t) = \mathcal{O}\left(h^n f^{(n)}(t) \right)$$

as desired.

E. Clarification of Amenability

The strange condition of being amenable of type (h, n) can be explained with an example. If $n = 5$ and $h = 10^{-4}$, which would be typical values, then we require the fairly modest requirements that

- $\left| f^{(6)}(t) \right| < v10^4 \left| f^{(5)}(t) \right|$
- $\left| f^{(7)}(t) \right| < v^2 10^8 \left| f^{(5)}(t) \right|$
- $\left| f^{(8)}(t) \right| < v^3 10^{12} \left| f^{(5)}(t) \right|$
- $\left| f^{(9)}(t) \right| < v^4 10^{16} \left| f^{(5)}(t) \right|$
- and so on…

Most any function that one can think of will satisfy this for $v = 1/2$. For example, even $f(t) = e^{100t}$ does not have its derivatives growing fast enough to fail to be amenable of type $(10^{-4}, 5)$. However, $f(t) = \sin\left(10^5 t\right)$ does fail this requirement, as does $f(t) = e^{10,001t}$. However, setting $t = 10^{-6}$ will repair both of these cases. Thus amenability is merely a question of setting h "sufficiently" small for a given $f(t)$ and degree of approximation n.

Functions arising from physical systems are unlikely to have this problem.