Journal of Computational and Applied Mathematics 382 (2021) 113041

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

On computational applications of the Levi-Civita field ☆

Darren Flynn [a], Khodr Shamseddine [b,*]

[a] Department of Mathematics & Statistics, University of Guelph, Guelph, Ontario N1G 2W1, Canada
[b] Department of Physics and Astronomy, University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada

ARTICLE INFO

Article history:
Received 31 December 2019
Received in revised form 30 May 2020

MSC:
26E30
12J25
11D88
46S10
65D25
65D30

Keywords:
Non-Archimedean analysis
Levi-Civita field
Numerical methods
Numerical differentiation
Numerical integration
Bernoulli numbers

ABSTRACT

In this paper, we study the computational applications of the Levi-Civita field whose elements are functions from the additive abelian group of rational numbers to the real numbers field, with left-finite support. After reviewing the algebraic and order structures of the Levi-Civita field, we introduce the Tulliotools library which implements the Levi-Civita field in the C++ programming language. We show that this software can replicate the results of (Shamseddine, 2015) by finding high order derivatives of certain functions faster than commercial software. We show how a similar method can be used to compute numerical sequences using generating functions and we compare this method with a number of conventional approaches. Finally, we show how the ability to quickly and accurately compute high order derivatives can be combined with Darboux's formula to preform numerical integration. We compare the performance of this new approach to numerical integration with more conventional approaches as well as commercial software and show promising results with regards to both speed and accuracy.

1. Motivation

Traditionally, physicists have used three fields to describe the universe, the rational numbers (denoted by $\mathbb{Q}$), the real numbers (denoted by $\mathbb{R}$), and the complex numbers (denoted by $\mathbb{C}$). These fields are **Archimedean fields** because they satisfy the so-called **Archimedean property**. The dominance of the Archimedean fields within physics is easily understood since the Archimedean property agrees with our intuitive understanding of distance. There are, however, many fields which fail to satisfy the Archimedean property and these are called **non-Archimedean valued fields**. Non-Archimedean valued fields have applications in computing the limits and asymptotic behaviour of analytic functions; the seminal work seems to be that of Lightstone and Robinson who, for example, was able to compute the incomplete factorial function to a high degree of precision by summing a finite number of terms of a divergent series [1]. In this paper we will focus on one particular non-Archimedean valued field called the Levi-Civita field, which has the useful property of being small enough to be implemented on a computer. The value of the Levi-Civita field from the perspective of computational applications is that it allows one to compute certain limits of real valued functions directly rather than by approximation. For example, given a differentiable function $f : I \subset \mathbb{R} \to \mathbb{R}$, the derivative of f at some point $x_0 \in I$ is given by

$$f'(x_0) = \lim_{h\to 0} \frac{f(x_0) - f(x_0 - h)}{h}.$$

☆ This research was funded by the Natural Sciences and Engineering Council of Canada (NSERC, Grant # RGPIN/4965-2017) and by the GETS program at the University of Manitoba.
* Corresponding author.
E-mail addresses: darrenfp@uoguelph.ca (D. Flynn), Khodr.Shamseddine@umanitoba.ca (K. Shamseddine).

https://doi.org/10.1016/j.cam.2020.113041

To compute this conventionally one would either use symbolic manipulation to reduce the expression so that it never involves division by h or one would choose h to be small enough that the error caused by its inclusion is less than some predetermined tolerance. The first possibility, while it produces accurate results, is unsatisfactory. Firstly, because symbolic manipulation is computationally slower than numerical calculations, and secondly, because it is not always clear what manipulations are necessary to produce the desired result. The second approach retains the speed of numerical computation but suffers from the issue that it is often difficult to determine how small h must be to ensure the result is sufficiently precise. Moreover, the numerical method is highly susceptible to rounding errors. It has been shown [2] that by employing the Levi-Civita field all of these issues can be addressed; we will discuss this further in Section 4. Our purpose in this paper is to investigate new computational applications of the Levi-Civita field; in the course of our investigation, we will also have the opportunity to compare our results with those from [2]. Similar computational applications are obtained with the numerical system employed by Sergeyev and his collaborators; see, for example, [3,4].

2. Introduction to the Levi-Civita field

In this section, we will present a brief review of the algebraic and topological structures of the Levi-Civita field $\mathcal{R}$. A more exhaustive survey of the recent research on the field is found in [5] and, unless otherwise stated, the reader may understand this to be the relevant reference throughout. We begin with a number of definitions.

Definition 1 (*Left-finite Subset of* $\mathbb{Q}$)**.** Let $A \subset \mathbb{Q}$ and suppose that for any $q \in \mathbb{Q}$, the set

$$A_{<q} := \{a \in A | a < q\}$$

is finite. Then we say that A is a **left-finite subset of** $\mathbb{Q}$.

Definition 2 (*The Support of a Function from* $\mathbb{Q}$ *to* $\mathbb{R}$)**.** Let $f : \mathbb{Q} \to \mathbb{R}$. Then **the support of f** is denoted by $\text{supp}(f)$ and is defined to be

$$\text{supp}(f) := \{q \in \mathbb{Q} | f(q) \neq 0\} .$$

Definition 3 (*The Set* $\mathcal{R}$)**.** We define

$$\mathcal{R} := \{f : \mathbb{Q} \to \mathbb{R} | \text{supp}(f) \text{ is left-finite}\} .$$

Elements of $\mathcal{R}$ are functions from $\mathbb{Q}$ to $\mathbb{R}$ and in the course of this paper we will have the need to discuss these elements evaluated at specific points in their domain as well as functions on $\mathcal{R}$. To avoid confusion we use the following notation.

Remark 4. We employ the convention that square brackets (i.e. '[' and ']') denote an element of $\mathcal{R}$ evaluated at some point in $\mathbb{Q}$ whereas curved brackets (i.e. '(' and ')') denote a function on $\mathcal{R}$ evaluated at a point in that set. So, for example, if we have $x \in \mathcal{R}$, $q \in \mathbb{Q}$, and $f : \mathcal{R} \to \mathcal{R}$, then

- $x[q] \in \mathbb{R}$ denotes an element of $\mathcal{R}$ evaluated at a point in $\mathbb{Q}$. The result of the evaluation will of course be a real number.
- $f(x) \in \mathcal{R}$ denotes a function evaluated at a point in $\mathcal{R}$. The result of the evaluation is another element of $\mathcal{R}$.
- $f(x)[q] \in \mathbb{R}$ denotes a function evaluated at a point in $\mathcal{R}$ and the result of that evaluation (itself an element of $\mathcal{R}$) evaluated at a point in $\mathbb{Q}$.

Definition 5. Let $x \in \mathcal{R}$ be given; then we define

$$\lambda(x) := \begin{cases} \min \text{supp}(x) & \text{if } x \neq 0 \\ \infty & \text{if } x = 0 \end{cases}$$

where ∞ is the same symbol used to obtain the extended real number system $\bar{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$ and it is used in the same way here as in the classical case. ∞ is not an element of the Levi-Civita field.

Note that, in the above definition, the minimum is guaranteed to exist by the left-finiteness of the support. In fact, for $x \in \mathcal{R}$, $\lambda(x)$ corresponds to the "order of magnitude" of x; we make this connection more rigorous below after we have defined the operations and order on $\mathcal{R}$.

Definition 6 (*Addition and Multiplication On* $\mathcal{R}$)**.** Suppose $x, y \in \mathcal{R}$; then we define for every $q \in \mathbb{Q}$

- $(x + y)[q] = x[q] + y[q]$
- $(x \cdot y)[q] = \sum_{\substack{q_1 \in \text{supp}(x) \\ q_2 \in \text{supp}(y) \\ q_1 + q_2 = q}} x[q_1] \cdot y[q_2]$

We have from [5] that if A, B are left-finite sets then so is $A+B$; moreover, if $r \in A+B$ then there are only finitely many pairs $(a, b) \in A \times B$ such that $a+b=r$. This fact ensures that multiplication on $\mathcal{R}$ is well defined since the sum in the definition of the multiplication will always have finitely many terms and hence it will always converge. Under these definitions of addition and multiplication $(\mathcal{R}, +, \cdot)$ is a field [5], and in fact we can isomorphically embed the real numbers into $\mathcal{R}$ as a subfield using the map $\Pi : \mathbb{R} \to \mathcal{R}$ defined by

$$\Pi(x)[q] := \begin{cases} x & \text{if } q = 0 \\ 0 & \text{if } q \neq 0. \end{cases}$$

Definition 7 (*Order on $\mathcal{R}$*)**.** Let $x, y \in \mathcal{R}$ be distinct. Then we say $x > y$ if $(x-y)[\lambda(x-y)] > 0$. We say $x < y$ if $y > x$ and we say $x \geq y$ if either $x = y$ or $x > y$.

Under this order relation $(\mathcal{R}, \geq)$ is a totally ordered field. Moreover, the embedding of $\mathbb{R}$ into this field via the map Π defined above is order preserving [5].

Definition 8 ($\ll, \gg, \sim, \approx,$ *and* $=_q$)**.** Let $x, y \in \mathcal{R}$ be positive. Then we say that x is infinitely larger than y and write $x \gg y$ if for every $n \in \mathbb{N}$, $x - ny > 0$; and we say that x is infinitely smaller than y and write $x \ll y$ if $y \gg x$. We say that x is infinitely large if $x \gg 1$ and we say it is infinitely small or infinitesimal if $x \ll 1$. Suppose that $\lambda(x) = \lambda(y) = \lambda_0$ then we write $x \sim y$; if in addition we have that $x[\lambda_0] = y[\lambda_0]$ then we write $x \approx y$. Finally, we write $x =_q y$ if $x[q'] = y[q']$ for all $q' \leq q$.

Notice in the above definition that $x \gg y$ whenever $\lambda(x) < \lambda(y)$; also since $\lambda(1) = 0$, x is infinitely large if $\lambda(x) < 0$ and x is infinitesimal if $\lambda(x) > 0$. The non-zero real numbers satisfy $\lambda(x) = 0$ as does the sum of a real number and an infinitesimal number. We define $\lambda(0) = \infty$ so that, for every $x \in \mathcal{R}$ with $x \neq 0$, we have that $\lambda(x) < \lambda(0)$.

Definition 9 (*The Number d*)**.** We define the element $d \in \mathcal{R}$ as follows: for every $q \in \mathbb{Q}$,

$$d[q] := \begin{cases} 1 & \text{if } q = 1 \\ 0 & \text{if } q \neq 1. \end{cases}$$

It follows that d is infinitesimal ($\lambda(d) = 1$); moreover, following from the definition of multiplication, we have that for any $n \in \mathbb{N}$,

$$d^n[q] := \begin{cases} 1 & \text{if } q = n \\ 0 & \text{if } q \neq n. \end{cases}$$

It follows that the same holds if we replace n in the equation above with any $r \in \mathbb{Q}$. In particular we have that

$$d^{-1}[q] := \begin{cases} 1 & \text{if } q = -1 \\ 0 & \text{if } q \neq -1. \end{cases}$$

Since $\lambda(d^{-1}) = -1$, d^{-1} is infinitely large, this is consistent with d being infinitesimal and in fact it allows the statement of an interesting inequality, namely for any $x \in \mathcal{R} \cap \mathbb{R}^+$ we have that

$$0 < d < x < d^{-1}.$$

Thus, in the Levi-Civita field, the positive real numbers are bounded both above and below by positive $\mathcal{R}$ numbers.

Functions on non-Archimedean fields often display properties that appear very different from those of real-valued functions on the real field $\mathbb{R}$. In particular it is possible to construct continuous functions that are not bounded on a closed interval, continuous and bounded functions that attain neither a maximum nor a minimum value on closed intervals, and continuous and differentiable functions with a derivative equal to zero everywhere on their domain which are nevertheless non-constant [2]. These unusual properties are a result of the total disconnectedness of these structures in the valuation topology [2,6]. Much work has been done in showing that power series and analytic functions on the Levi-Civita field have the same smoothness properties as real power series and real analytic functions [7]. The effort to extend these properties to as large a class of functions as possible has been aided considerably by the introduction of the so-called weak topology on the Levi-Civita field which is strictly weaker than the order topology and thus allows for more power series to converge than the order topology. Here we briefly review the properties of power-series and analytic functions on the Levi-Civita field.

Definition 10. We say that a sequence in $\mathcal{R}$ **converges strongly** if it converges with respect to the order topology.

The "weak topology" mentioned above is constructed using the family of semi-norms defined below.

Definition 11 (*A Family of Semi-Norms on $\mathcal{R}$*)**.** For every $r \in \mathbb{R}$ define the map $\|\cdot\|_{(w,r)} : \mathcal{R} \to \mathbb{R}$ by

$$\|x\|_{(w,r)} := \max\left\{|x[q]| \,|\, q \in \operatorname{supp} x \cap (-\infty, r]\right\}.$$

Since every $x \in \mathcal{R}$ has a left-finite support, the maximum in the above definition is guaranteed to exist in $\mathbb{R}$.

Definition 12. Let $(s_n)_{n\in\mathbb{N}}$ be a sequence in $\mathcal{R}$. Then we say that this sequence **converges weakly** if there is an $s \in \mathcal{R}$ such that for every $\epsilon > 0$ in $\mathbb{R}$ there is a $N \in \mathbb{N}$ such that for every $m \geq N$,

$$\|s_m - s\|_{\left(w,\frac{1}{\epsilon}\right)} < \epsilon.$$

Convergence of sequences and series in both the order topology and weak topology was studied in details in [6]. In particular, convergence criteria for sequences and series were stated and proved. As a consequence of the weak convergence criterion for power series, it was shown that a convergent real power series with real radius of convergence η can be extended to the Levi-Civita field for any $x \in \mathcal{R}$ satisfying $|x| < \eta$ and $|x| \not\approx \eta$. This allows for the continuation to $\mathcal{R}$ of all convergent real power series; in particular, we can use the power series for the real trigonometric functions and the exponential function to define the continuations of those functions to $\mathcal{R}$ for any $x \in \mathcal{R}$ that is not infinitely large in absolute value. This turned out to be of great importance for the computational applications that we will discuss in the next section.

Finally, we note in passing here that it was shown in [6,8–10] that power series over $\mathcal{R}$ and the so-called $\mathcal{R}$-analytic functions (that is, functions that are given locally by weakly convergent power series) have the same smoothness properties as real power series and real analytic functions. In particular, they satisfy the intermediate value theorem, the extreme value theorem, the mean value theorem and the inverse function theorem; they are infinitely often differentiable; and they have unique antiderivatives (modulo a constant) within the family of $\mathcal{R}$-analytic functions. Hence, $\mathcal{R}$-analytic functions have been used as the building blocks for a Lebesgue-like integration theory [11] in which the integral satisfies similar properties to those of the Lebesgue integral of real Analysis.

3. The Tulliotools software

To use the Levi-Civita field in computational applications we will first develop a code that will allow a computer to operate on these numbers. The code that has been used in previous papers on this topic (COSY INFINITY) [2] is not easily accessible and so we constructed our own software for this purpose. Our code forms a static library in the C++ programming language and we tentatively name it Tulliotools in honour of the Italian mathematician Tullio Levi-Civita who first discovered the field that bears his name [12]. Tulliotools was created using **Microsoft Visual Studio 2015 Community Edition** and was compiled using default settings. The Tulliotools library defines how a computer can store an element of the Levi-Civita field (up to some specific depth) and defines the operations of addition, multiplication, and inversion. Subtraction and division are defined by addition of the additive inverse and multiplication with the multiplicative inverse, respectively. Tulliotools also includes the basic trigonometric and inverse trigonometric functions, the hyperbolic trigonometric functions, the exponential function, the natural logarithm, and the n^{th} root for an arbitrary integer n. When computing elementary functions we wish to employ the language's built-in functions as much as possible both for the sake of speed and accuracy. To accomplish this, we use the addition theorems for the aforementioned elementary functions. We then use the built-in functions to compute the contribution from the real part of the argument and we use Taylor series to compute the contribution from the infinitesimal part. For example,

$$\sin_{lc}(x) = \sin_{lc}(x_r + x_i) = \sin_r(x_r)\cos_t(x_i) + \cos_r(x_r)\sin_t(x_i),$$

where we have used the convention that for a given real analytic function, $f_{lc} : \mathcal{R} \to \mathcal{R}$ is the non-Archimedean continuation of the function, $f_r : \mathbb{R} \to \mathbb{R}$ is the real (built-in) function, $f_t : \mathcal{R} \to \mathcal{R}$ is the Taylor expansion (up to whatever depth is required) of the function, x_r is the real part of the argument (i.e. $x_r = x[0]$), and x_i is the infinitesimal part of the argument (i.e. $x_i = x - x[0]$). Computing the inverse trigonometric functions is more difficult than computing their trigonometric counterparts because they lack convenient addition theorems. Instead we make use of integration (which we discuss in a later section) and the fact that the derivatives of these functions are well known. For example, using the same convention as above, we have for $x_i \geq 0$ that

$$\begin{aligned}\arcsin_{lc}(x) &= \int_{t\in(0,x)} \frac{1}{\sqrt{1-t^2}} \\ &= \int_{t\in(0,x_r)} \frac{1}{\sqrt{1-t^2}} + \int_{t\in(x_r,x_r+x_i)} \frac{1}{\sqrt{1-t^2}} \\ &= \arcsin_r(x_r) + \int_{t\in(x_r,x_r+x_i)} \frac{1}{\sqrt{1-t^2}}.\end{aligned}$$

On the other hand, for $x_i < 0$, we have that

$$\begin{aligned}\arcsin_{lc}(x) &= \int_{t\in(0,x)} \frac{1}{\sqrt{1-t^2}} \\ &= \int_{t\in(0,x_r)} \frac{1}{\sqrt{1-t^2}} - \int_{t\in(x_r+x_i,x_r)} \frac{1}{\sqrt{1-t^2}} \\ &= \arcsin_r(x_r) - \int_{t\in(x_r+x_i,x_r)} \frac{1}{\sqrt{1-t^2}}.\end{aligned}$$

Notice that the only integrals that we actually need to compute are all over an infinitesimal interval, this allows us to compute them exactly (up to a given depth) by integrating the Taylor series of the integrand.

4. Numerical computation of derivatives

The first thing we would like to do with our newly developed library is to explore the applications to the numerical computation of derivatives developed in [2] and [13]. We begin by restating a number of definitions in our own notation and reviewing the underlying mathematical theory.

Definition 13 (*Computer Function*)**.** Let I be the set of all functions intrinsic to the C++ programming language as well as their inverse functions and the step function $s:\mathbb{R}\to\mathbb{R}$ defined by

$$s(x) := \begin{cases} 0 & \text{if } x \le 0 \\ 1 & \text{if } x > 0. \end{cases}$$

We define a **computer function** to be any real-valued function that can be obtained preforming a finite number of arithmetic operations and compositions using functions in I.

It is possible to extend computer functions to $\mathcal{R}$ using the extensions of power series with purely real coefficients as well as the step function, $x^{\frac{1}{n}}$, and $\frac{1}{x}$ [13].

Definition 14 (*Extendable Computer Function*)**.** Let f be a computer function, let $x_0 \in \mathbb{R}$ be in the domain of f, and let $x \in \mathcal{R}$. Then we say that f is extendable to $x_0 + x \in \mathcal{R}$ if $x_0 + x$ is in the domain of the extension of f to $\mathcal{R}$.

Definition 15 (*Continuation of Computer Functions to* $\mathcal{R}$)**.** It is shown in [13] that if f is a real computer function, x_0 is in the domain of f, and f is extendable to $x_0 \pm d$ then there is a $\eta > 0$ in $\mathbb{R}$ such that for $x \in \mathbb{R}$ with $0 < x < \eta$

$$f(x_0 \pm x) = \sum_{i=0}^{\infty} a_i^{\pm} x^i + \sum_{j=1}^{j^{\pm}} x^{q_j^{\pm}} R_j^{\pm}(x) \tag{1}$$

where for all $j \in \{1,\ldots,j^{\pm}\}$, $R_j^{\pm}$ is a power series with $R_j^{\pm}(0) \neq 0$ and with a radius of convergence at least as large as η, and $q_j^{\pm}$ are nonzero rational numbers that are not positive integers. Since the right hand side of Eq. (1) contains only roots, negative integer powers, and power series (for which we have already defined a continuation to $\mathcal{R}$) we may define the continuation of f to $x_0 \pm x \in \mathcal{R}$ such that $0 < x < \eta$ and $x[0] \neq \eta$ by

$$\bar{f}(x_0 \pm x) := \sum_{i=0}^{\infty} a_i^{\pm} x^i + \sum_{j=1}^{j^{\pm}} x^{q_j^{\pm}} \bar{R}_j^{\pm}(x)$$

where $\bar{R}_j^{\pm}$ is the continuation of $R_j^{\pm}$ to $\mathcal{R}$.

Now suppose that f is a real computer function defined at $x_0 \in \mathbb{R}$ and extendable to $x_0 \pm d$. Then we have that

$$\bar{f}(x_0 \pm d) = \sum_{i=0}^{\infty} a_i^{\pm} d^i + \sum_{j=1}^{j^{\pm}} d^{q_j^{\pm}} \bar{R}_j^{\pm}(d).$$

The equation above entails that for any $n \in \mathbb{N}$

$$(\bar{f}(x_0 \pm d))[n] = \left(\sum_{i=0}^{\infty} a_i^{\pm} d^i + \sum_{j=1}^{j^{\pm}} d^{q_j^{\pm}} \bar{R}_j^{\pm}(d) \right)[n]$$

However if it happens that for some $m \in \mathbb{N}$, f is m times differentiable at x_0, then we must have that $q_j^{\pm} > m$ for every $j \in \{1,\ldots,j^{\pm}\}$ and $a_i^{+} = (-1)^i a_i^{-} = \frac{f^{(i)}(x_0)}{i!}$ for every $i \in \{1,\ldots,m\}$ [13]. Hence we have that

$$\bar{f}(x_0 + d) =_m \sum_{i=0}^{m} a_i^{+} d^i$$

and

$$\bar{f}(x_0 - d) =_m \sum_{i=0}^{m} a_i^{-} d^i$$

with $a_i^{-} = (-1)^i a_i^{+}$ in which case we have that for any $i \in \{1,\ldots,m\}$

$$i!a_i^{+} = i!\bar{f}(x_0 + d)[i] = f^{(i)}(x_0) = (-1)^i i!\bar{f}(x_0 - d)[i] = (-1)^i i! a_i^{-}.$$

Table 1
First 13 derivatives of g as computed by Mathematica.

n	$g^{(n)}(0)$	Time (s)	$g^{(n)}(1)$	Time (s)	$g^{(n)}(2)$	Time (s)
0	0	0.0002	0.837192955627	0.308	0.888584820075	0.196
1	1.26027064058	0.002	0.407172848084	0.054	−0.317934898588	0.009
2	0	0.003	−0.618746127149	0.014	−0.651895577342	0.014
3	−5.35211351959	0.03	0.0122192107521	0.062	0.416693615024	0.109
4	0	0.112	−4.31613114141	0.171	−1.64786996410	0.178
5	121.167674235	0.329	15.652	0.542463	−19.6728802712	0.838
6	0	0.953	78.5779028176	1.747	−20.2596967220	2.106
7	−5627.09443507	3.0960	−685.282937503	5.835	615.708023511	6.997
8	0	10.691	−1285.70479011	19.589	2622.42370100	21.9916
9	429913.385688	32.896	16481.3309024	57.559	−30298.4169665	61.748
10	0	91.505	227724.788971	153.641	−114129.369772	161.231
11	−49831093.1255	238.66	−257502.130656	397.697	3525304.24927	399.671
12	0	583.886	Aborted	>1 h	Aborted	>1 h
13	8083947834.90	1623.43	Aborted	>1 h	Aborted	>1 h

It is possible to make an argument similar to the above but in the opposite direction which allows for the following theorem from [13].

Theorem 16. *Let f be a computer function that is continuous at x_0 and extendable to $x_0 \pm d$. Then f is m times differentiable at x_0 if and only if*

$$\bar{f}(x_0 + d) =_m \sum_{i=0}^{m} a_i^+ d^i,$$

and

$$\bar{f}(x_0 - d) =_m \sum_{i=0}^{m} a_i^- d^i$$

with $a_i^- = (-1)^i a_i^+$ for $i \in \{1, \ldots, m\}$. Moreover, in this case

$$i!a_i^+ = i!\bar{f}(x_0 + d)[i] = f^{(i)}(x_0) = (-1)^i i!\bar{f}(x_0 - d)[i] = (-1)^i i!a_i^-.$$

for all $i \in \{1, \ldots, m\}$.

Theorem 16 gives us a method to both check the differentiability and numerically compute the derivatives of real computer functions and it was used to great effect in [2,13]. Below we replicate the success of that paper using the Tulliotools library and we produce some additional examples. As in that paper, we compare our results against **Wolfram Mathematica 11.3**. It is worth noting that, by the nature of the software, Tulliotools computes all (up to a given depth) derivatives simultaneously whereas Mathematica computes them each individually. This difference has no significant effect on our conclusions, however, because Mathematica takes significantly longer than Tulliotools to compute higher order derivatives. Even if we generously assume that Mathematica could, in the time it takes to compute the n^{th} derivative, compute the first $(n-1)$ derivatives as well then the above method still easily out-performs it. Indeed, for the sake of time, we aborted Mathematica's calculations wherever they lasted for longer than an hour. The time to compute the results was found for Mathematica using the built-in **Absolutetiming** function and for Tulliotools using the **Chrono** libraries' **high precision clock** function. We use the following two functions to test the relative ability of our software to compute derivatives.

$$g(x) := \frac{\sin(\sin(\sin(\sin(\sin(x)))))}{\cos(\cos(\cos(\cos(\cos(x)))))}$$

$$h(x) := \frac{\sin\left(x^3 + 2x + 1\right) + \frac{3+\cos(\sin(\ln|1+x|))}{\exp\left(\tanh\left(\sinh\left(\cosh\left(\frac{\sin(\cos(\tan(\exp(x))))}{\cos(\sin(\exp(\tan(x+2))))}\right)\right)\right)\right)}}{2 + \sin\left(\sinh\left(\cos\left(\arctan\left(\ln\left(\exp x + x^2 + 3\right)\right)\right)\right)\right)}$$

The function g provides us with an intermediate challenge and already we can see Mathematica falling behind Tulliotools for higher order derivatives. (See Tables 1 and 2.)

Table 2
First 14 derivatives of g as computed by Tulliotools.

n	$g^{(n)}(0)$	$g^{(n)}(1)$	$g^{(n)}(2)$
0	0	0.837192955627	0.888584820075
1	1.26027064058	0.407172848084	−0.317934898588
2	0	−0.618746127149	−0.651895577342
3	−5.35211351959	0.0122192107521	0.416693615024
4	0	−4.31613114141	−1.6478699641
5	121.167674235	15.6517446222	−19.6728802712
6	0	78.5779028176	−20.259696722
7	−5627.09443507	−685.282937503	615.708023511
8	0	−1285.70479011	2622.423701
9	429913.385688	16481.3309024	−30298.4169665
10	0	227724.788971	−114129.369772
11	−49831093.1255	−257502.130656	3525304.24927
12	0	−37912424.234	4688958.30662
13	8083947834.9	−13666350.8705	−495861347.515
14	0	4734886537.81	−152712264.273
Total (s)	0.207	0.193	0.218

Table 3
First 8 derivatives of h as computed by Mathematica.

n	$h^{(n)}(0)$	Time (s)	$h^{(n)}(1)$	Time (s)	$h^{(n)}(5)$	Time (s)
0	1.00484531901	0.16	0.268357844508	0.10	0.283393816437	0.30
1	0.460143808963	0.15	−1.44525348415	0.04	12.1382777290	0.14
2	−5.26609756823	0.30	7.31608659872	0.14	28594.4371105	0.13
3	−52.8216335199	0.70	40.8666551717	0.32	10161444.9755	0.31
4	−108.468284784	1.67	404.249076373	1.13	−32567374548.9	1.59
5	16451.4428641	4.73	−5092.63654924	4.53	$-1.29110802579\times10^{14}$	5.70
6	541334.997022	21.11	−19854.7155232	28.96	$-2.98281735849\times10^{17}$	35.47
7	794864118.936	124.35	1611673.41227	171.20	$-4.20384900033\times10^{20}$	184.15
8	−144969388.210	787.34	−86895133.1031	2426.91	Aborted	>1 h

Table 4
First 14 derivatives of h as computed by Tulliotools.

n	$h^{(n)}(0)$	$h^{(n)}(1)$	$h^{(n)}(5)$
0	1.00484531901	0.268357844508	0.283393816437
1	0.460143808963	−1.44525348415	12.138277729
2	−5.26609756823	7.31608659872	28594.4371105
3	−52.8216335199	40.8666551717	10161444.9755
4	−108.468284784	404.249076373	−32567374548.9
5	16451.4428641	−5092.63654924	$-1.29110802579\times10^{14}$
6	541334.997022	−19854.7155232	$-2.98281735849\times10^{17}$
7	7948641.18936	1611673.41227	$-4.20384900033\times10^{20}$
8	−144969388.21	−86895133.1031	$2.78479886876\times10^{23}$
9	−15395959663	3193445289.11	$4.77510276588\times10^{27}$
10	−618406836695	−90967229524	$2.1329279112\times10^{31}$
11	$-1.17903146156\times10^{13}$	$1.74199571026\times10^{12}$	$6.24639715614\times10^{34}$
12	$4.03355397865\times10^{14}$	$1.49155784151\times10^{13}$	$9.55133940595\times10^{37}$
13	$5.51065265978\times10^{16}$	$-3.85982238753\times10^{15}$	$-2.68590144823\times10^{41}$
14	$3.27278740268\times10^{18}$	$2.59042564116\times10^{17}$	$-3.11629245228\times10^{45}$
Time (s)	0.826	0.505	0.580

Finally we have function h which was obtained from [2,13]. Here Mathematica is slower than Tulliotools even for low order derivatives and it was unable to find any derivatives past the eighth in less than an hour. (See Tables 3 and 4.)

5. Numerical computation of Bernoulli numbers

Evaluating analytic functions with real coefficients at infinitesimal points can do more than finding the derivatives of the function, it also allows us to calculate sequences of numbers defined by a generating function. Consider for example the following definition of the Bernoulli numbers.

Definition 17 (*Bernoulli Numbers*)**.** The **Bernoulli numbers** are precisely those numbers $B_n \in \mathbb{R}$ such that for any $t \in \mathbb{R}$

$$\frac{t}{e^t - 1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} t^n.$$

The Bernoulli numbers are usually calculated using either the summation formula or the recursive formula.

$$B_n = \sum_{i=0}^{n} \sum_{j=0}^{i} (-1)^j \binom{i}{j} \frac{j^n}{i+1}$$

$$B_n = \delta_{n,0} - \sum_{i=0}^{n-1} \binom{n}{i} \frac{B_i}{n-i+1}$$

Using the Levi-Civita numbers, however, we are able to calculate the Bernoulli numbers directly from the generating function. Notice that

$$\left(\frac{d}{e^d - 1} \right) = \sum_{n=0}^{\infty} \frac{B_n}{n!} d^n$$

from which we find that

$$B_n = n! \left(\frac{d}{e^d - 1} \right) [n].$$

In fact there are a number of different ways we can calculate the Bernoulli numbers along the same lines. Recall that, for $t \in \mathbb{R}$ with $|t| < \frac{\pi}{2}$, we have

$$\tan(t) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} 2^{2n} (2^{2n} - 1) B_{2n}}{(2n)!} t^{2n-1}.$$

This identity can be extended to $\mathcal{R}$ and holds for any $t \in \mathcal{R}$ satisfying $|t| < \frac{\pi}{2}$ and $\frac{\pi}{2} - |t| \sim 1$ [6]. It follows that

$$B_{2n} = \frac{(-1)^{n-1} (2n)!}{2^{2n} (2^{2n} - 1)} \left(\tan(d) \right) [2n - 1]$$

for all $n \in \mathbb{N}$. One might object that this equation only allows the calculation of every other Bernoulli number; however, aside from $B_1 = -\frac{1}{2}$, every odd Bernoulli number is zero anyway.

A glance at Table 5 will show that the first three methods displayed produce incorrect values for the odd Bernoulli numbers. This is caused by "rounding error". The problem of rounding errors can be overcome with the use of a so-called "arbitrary precision library" and it would be interesting to compare the computation time of these methods with the use of such a library.

6. Methods of numerical integration

The ability to compute high-order derivatives of analytic functions allows for some interesting strategies for numerical integration; in particular, we have **Darboux's Formula** which allows us to approximate an integral using our knowledge of the integrand and its derivatives at the end points of the interval of integration [14].

Proposition 18 (*Darboux's Formula for the Levi-Civita Field*)**.** *Let $a, b \in \mathcal{R}$ satisfy $a < b$ and let $f : \mathcal{R} \to \mathcal{R}$ be an analytic function on the interval $[a, b]$. Suppose $\phi : \mathcal{R} \to \mathcal{R}$ is a polynomial of degree n, then we have that*

$$\sum_{m=0}^{n} (-1)^m (b-a)^m \left[\phi^{(n-m)}(1) f^{(m)}(b) - \phi^{(n-m)}(0) f^{(m)}(a) \right]$$
$$= (-1)^n (b-a)^{n+1} \int_{t \in [0,1]} \phi(t) f^{(n+1)}(a + t(b-a)).$$

Proof. As in the real Analysis case, this identity can be proven by repeated integration by parts. □

Rearranging the terms in Darboux's formula yields the relation

$$\phi^{(n)}(1) f(b) - \phi^{(n)}(0) f(a) = (-1)^n (b-a)^{n+1} \int_{t \in [0,1]} \phi(t) f^{(n+1)}(a + t(b-a))$$
$$- \sum_{m=1}^{n} (-1)^m (b-a)^m \left[\phi^{(n-m)}(1) f^{(m)}(b) - \phi^{(n-m)}(0) f^{(m)}(a) \right].$$

Table 5
Bernoulli Numbers computed in various ways.

n	Additive formula	Recursive formula	Generating function	tan Formula	Exact (to six decimals)	
0	1	1	1	–	1	
1	-0.5	-0.5	-0.5	–	-0.5	
2	0.166667	0.166667	0.166667	0.166667	0.166667	
3	0	0	0	0	0	
4	-0.0333333	-0.0333333	-0.0333333	-0.0333333	-0.0333333	
5	0	0	0	0	0	0
6	0.0238095	0.0238095	0.0238095	0.0238095	0.0238095	
7	2.23517×10^{08}	0	-3.14748×10^{12}	0	0	
8	0.0757571	0.0757576	0.0757576	0.0757576	0.0757576	
9	3.8147×10^{06}	-6.10623×10^{16}	-1.73112×10^{11}	0	0	
10	-0.252197	-0.253114	-0.253114	-0.253114	-0.253114	
11	-0.0078125	1.11022×10^{15}	2.70054×10^{09}	0	0	
12	1.125	1.16667	1.16667	1.16667	1.16667	
13	-32	-3.73035×10^{14}	1.84312×10^{06}	0	0	
14	-256	-7.09216	-7.09216	-7.09216	-7.09216	
15	98304	1.42109×10^{14}	-0.000173537	0	0	
16	-3.93216×10^{06}	54.9712	54.966	54.9712	54.9712	
17	1.24151×10^{09}	-3.18323×10^{12}	0.0783084	0	0	
18	7.62357×10^{10}	-529.124	-530.399	-529.124	-529.124	
19	3.43597×10^{11}	2.09184×10^{11}	-1.38482	0	0	
20	-3.40849×10^{14}	6192.12	5976.59	6192.12	6192.12	
21	1.28071×10^{16}	-3.7835×10^{10}	98.5388	0	0	
22	-1.97258×10^{18}	-86580.3	80473.3	-86580.3	-86580.3	
23	-1.29704×10^{20}	4.42378×10^{09}	-3.88571×10^{06}	0	0	
24	5.27577×10^{21}	1.42552×10^{06}	1.41551×10^{07}	1.42552×10^{06}	1.42552×10^{06}	
25	1.01531×10^{23}	-7.82311×10^{08}	5.59158×10^{08}	0	0	
26	-7.20822×10^{25}	-2.72982×10^{07}	1.14962×10^{10}	-2.72982×10^{07}	-2.72982×10^{07}	
27	6.50886×10^{27}	1.3113×10^{06}	-3.67793×10^{11}	0	0	
28	-9.06172×10^{29}	6.01581×10^{08}	6.06192×10^{12}	6.01581×10^{08}	6.01581×10^{08}	
29	-4.69823×10^{31}	5.72205×10^{06}	-5.40877×10^{13}	0	0	

However, because ϕ is a polynomial of degree n, we have that $\phi^{(n)}(0) = \phi^{(n)}(1) = \phi_0$ and hence

$$\phi^{(n)}(1)f(b) - \phi^{(n)}(0)f(a) = \phi_0 \int_{t\in[a,b]} f'(t).$$

Thus, substituting this last equality into Darboux's formula and then replacing f' with f in the formula, we obtain:

$$\int_{t\in[a,b]} f(t) = \frac{1}{\phi_0}\sum_{m=1}^{n}(-1)^{m+1}(b-a)^m\left[\phi^{(n-m)}(1)f^{(m-1)}(b) - \phi^{(n-m)}(0)f^{(m-1)}(a)\right] + \frac{1}{\phi_0}(-1)^n(b-a)^{n+1}\int_{t\in[0,1]}\phi(t)f^{(n)}(a+t(b-a)). \tag{2}$$

So we can integrate f by finding its derivatives as well as the derivatives of ϕ, the integral term on the right hand side of Eq. (2) is our error. Different choices of ϕ will reduce Eq. (2) to different summation formulas; for example, if ϕ is the nth degree Bernoulli polynomial then Eq. (2) is equivalent to the Euler–Maclaurin equation. Similarly, if ϕ is $(t-1)^n$ or t^n then the right hand side of Eq. (2) goes to the Taylor series of the integrand about the left or right endpoint as $n \to \infty$ [14]. Another polynomial we investigate is $\prod_{i=1}^{n}(t - \frac{i}{n+1})$ with the idea that, even if the n^{th} derivative of the integrand is large, the frequent sign changes in ϕ will cause the integral term on the right hand side of Eq. (2) to be small. Although Darboux's formula can be made equivalent to the Taylor expansion of the integrand about an end point, the same is not possible for arbitrary points in the interval of integration. For this reason we also experiment with integrating directly by a Taylor series about the midpoint of the interval of integration. First we integrate a selection of high order polynomials only evaluating them in the infinitesimal neighbourhood about the end points (or the midpoint as the case may be). For comparison we integrate the same polynomials using the Trapezoidal Rule and Simpson's Rule in the normal way (i.e. without use of infinitesimals). We also compute the integrals symbolically using Mathematica and, where it is possible, SymbolicC++. In addition to its symbolic integration method, Mathematica also provides a method of numerical integration; in fact, this method does not correspond to any single integration technique but instead selects from a number of different techniques depending on the specific integral in question. In principle, as long as the degree of the polynomial to be integrated is less than the depth to which we can find its derivatives (which is to say the depth of calculation minus 1), our methods should produce an exact answer. We consider both this case and the case where the degree of the integrand is greater than our depth of calculation (which happens to be 25 for this experiment). The

Table 6
Integral of $P_{10}Q_5$ from 0 to 10.

Method of computation	Result	Time (s)
Central point Taylor series	$5.60845249807 \times 10^{13}$	0.0011
Darboux's Formula (Bernoulli polynomials)	$5.60845249842 \times 10^{13}$	0.0249
Darboux's Formula (Frequent sign change polynomial)	$5.60845249807 \times 10^{13}$	0.0060
Simpsons rule (1000 steps)	$5.47069797559 \times 10^{13}$	0.6372
Trapezoidal rule (1000 steps)	$5.50529044403 \times 10^{13}$	0.6114
Mathematica (symbolic)	$5.60845249807 \times 10^{13}$	4.3917
Mathematica (numeric)	$5.60845249807 \times 10^{13}$	0.1592

Table 7
Integral of $P_{12}Q_8$ from 0 to 10.

Method of computation	Result	Time (s)
Central point Taylor series	$1.41666070259 \times 10^{18}$	0.0021
Darboux's Formula (Bernoulli polynomials)	$1.41666067675 \times 10^{18}$	0.0224
Darboux's Formula (Frequent sign change polynomial)	$1.41666070259 \times 10^{18}$	0.0111
Simpsons rule (1000 steps)	$1.37030525479 \times 10^{18}$	0.8658
Trapezoidal rule (1000 steps)	$1.38196505555 \times 10^{18}$	1.0181
Mathematica (symbolic)	$1.41666070259 \times 10^{18}$	6.5037
Mathematica (numeric)	$1.41666070259 \times 10^{18}$	0.2105

following two families of polynomials provide us with an ample supply of integrands to test. Let $n \in \mathbb{N}$ be given. Then we define

- A polynomial of degree n

$$P_n(x) := \prod_{i=1}^{n}(x - \frac{i}{\pi})$$

- The Bernoulli polynomial of degree n, where B_i is the i^{th} Bernoulli number, given by

$$Q_n(x) := \sum_{i=0}^{n} \binom{n}{i} B_{n-i} x^i$$

As Tables 6 and 7 show, non-Archimedean methods of integration produced good numerical values for the given integrals and reliably did so faster than Mathematica; this suggests that non-Archimedean methods provide an advantage when it comes to integrating polynomials. It is also interesting to investigate the performance of non-Archimedean methods when integrating analytic functions; to that end, we obtained a selection of analytic functions from [15]. In this case we break the interval of integration into smaller steps to ensure that the error terms involved do not diverge, the depth of calculation remains 15 throughout. For comparison, we compute these same integrals using Mathematica both symbolically and numerically. Table 8 below shows the results regarding the function $f(x) := \frac{x^2}{\sin^2(x)}$ from 0 to $\frac{\pi}{4}$, these results are typical in that the non-Archimedean methods of integration are competitive with Mathematica's symbolic method of integration but were slower than its numeric method.

To conclude, we would like to determine if there is any class of functions which Tulliotools is better at integrating than Mathematica, and in fact, it seems that there is: Mathematica has a well-known difficulty integrating highly oscillatory functions [16]. Since Tulliotools has access to not only the value of the integrand but also its derivatives it should be substantially better at integrating highly oscillatory functions. To test this, we consider the function

$$g(x) := P_5(\cos(100x))Q_5(\cos(100x)).$$

As may be seen in Table 9, when Mathematica numerically integrates this function under default settings it produces an incorrect answer; Tulliotools, on the other hand, is able to attain the first four digits correctly in approximately half the time Mathematica takes. The depth of calculation is 5.

The results in this section are hardly conclusive; in particular a fair comparison would require that Tulliotools be able to actively adjust how the interval of integration is partitioned based on the specific integrand. Moreover, a more sophisticated method of comparison will be necessary to ensure that Mathematica and Tulliotools are always competing to attain the same degree of precision. Nevertheless, these results are sufficient to establish that non-Archimedean methods of numerical integration are highly versatile and have the potential to improve upon conventional methods.

Table 8
Integral of f from 0 to $\frac{\pi}{4}$ with various step sizes.

Method of computation	10 steps	Time (s)	50 steps	Time (s)	100 steps	Time (s)	500 steps	Time (s)	1000 steps	Time (s)
Central point Taylor series	0.938429184613	0.215	0.862808426244	0.709	0.853180614292	1.280	0.845448903068	5.241	0.844480579999	10.446
Darboux's Formula (Bernoulli polynomials)	0.843511841685	0.238	0.843511841685	1.706	0.843511841685	4.014	0.843511841685	13.269	0.843511841685	26.723
Darboux's Formula (Frequent sign change polynomial)	0.843511841685	0.208	0.843511841685	1.580	0.843511841685	2.317	0.843511841685	11.468	0.843511841685	22.712

Method of computation	Integral	Time (s)
Mathematica (symbolic)	0.843511841685	6.420
Mathematica (numeric)	0.843511841685	0.069
Exact value	0.843511841685	n/a

Table 9
Integral of g from 0 to 10.

Method of computation	1000 steps	Time (s)	5000 steps	Time (s)	10000 steps	Time (s)
Central point Taylor series	186.369934703	2.821	186.372340098	14.669	186.372340207	25.436123
Darboux's Formula (Bernoulli polynomials)	186.389214786	6.433	186.372340097	26.562	186.372340207	52.864
Darboux's Formula (Frequent sign change polynomial)	186.365945702	4.663	186.372340097	24.594	186.372340207	46.185
Method of computation	Integral			Time (s)		
Mathematica (symbolic)	186.372340189			4324.78		
Mathematica (numeric with adjusted settings)	186.372340188			66.96		
Mathematica (numeric with default settings)	182.322			5.964		

7. Research outlook

The results presented in this paper allow for numerous possibilities for future research. On the theoretical side of things, one might start by determining the computational complexity of various operations on the Levi-Civita field, and comparing them to their classical counterparts. More practically, the C++ implementation of the Levi-Civita field could be expanded to allow it to operate in conjunction with an arbitrary precision library. That would be helpful in the study of generating functions and numeric sequences, as it would reduce the effect of rounding errors. There are also many ideas to pursue with regards to numerical integration. We would like to expand the C++ implementation, so that it is capable of actively adjusting parameters in response to the various numerical approximations produced. We are also interested in implementing Monte Carlo methods of integration, as these might make better use of the ability to compute high order derivatives of the integrand. One key mathematical problem related to this topic is finding a way to easily approximate the radius of convergence of the power series representation of an integrand. Here too, we hope to benefit from our ability to compute high order derivatives of the integrand. Finally, it has been shown that it is possible to construct a rigorous delta function on the Levi-Civita field and that the resulting non-Archimedean delta function can be used to solve ordinary differential equations [17,18]. It would be interesting to implement that method as a computer program and explore its potential as compared to other numerical methods for solving ODEs.

References

[1] A.H. Lightstone, A. Robinson, Nonarchimedean Fields and Asymptotic Expansions, North-Holland Pub. Co., 1975.
[2] K. Shamseddine, Analysis on the Levi–Civita field and computational applications, Appl. Math. Comput. 255 (2015) 44–57.
[3] Ya.D. Sergeyev, Higher order numerical differentiation on the Infinity computer, Optim. Lett. 5 (2011) 575–585.
[4] F. Iavernaro, F. Mazzia, M.S. Mukhametzhanov, Ya.D. Sergeyev, Conjugate-symplecticity properties of Euler–Maclaurin methods and their implementation on the Infinity Computer, Appl. Numer. Math. 155 (2020) 58–72.
[5] K. Shamseddine, New Elements of Analysis on the Levi–Civit Field (Ph.D. thesis), Michigan State University, Department of Mathematics and Department of Physics & Astronomy, 1999.
[6] K. Shamseddine, M. Berz, Convergence on the Levi–Civita field and study of power series, in: Proceedings of the Sixth International Conference on P-Adic Analysis, in: Lecture Notes in Pure and Applied Mathematics, Marcel Dekker, 2000, pp. 283–299.
[7] K. Shamseddine, A brief survey of the study of power series and analytic functions on the Levi–Civita fields, Contemp. Math. 596 (2013) 269–279.
[8] K. Shamseddine, M. Berz, Analytical properties of power series on Levi–Civita fields, Ann. Math. Blaise Pascal 12 (2) (2005) 309–329.
[9] K. Shamseddine, M. Berz, Intermediate value theorem for analytic functions on a Levi–Civita fields field, Bull. Belg. Math. Soc. Simon Stevin 14 (2007) 1001–1015.
[10] K. Shamseddine, Absolute and relative extrema, the mean value theorem and the inverse function theorem for analytic functions on a Levi–Civita fields field, Contemp. Math. 551 (2011) 257–268.
[11] K. Shamseddine, M. Berz, Measure theory and integration on the Levi–Civita fields field, Contemp. Math. 319 (2003) 369–387.
[12] T. Levi-Civita, Sugli infiniti ed infinitesimali actuali quali elementi analytici, Opere Matematiche 1 (1893–1900).
[13] K. Shamseddine, M. Berz, The differential algebraic structure of the Levi–Civita fields field and applications, Int. J. Appl. Math. 3 (4) (2000) 449–464.
[14] M.G. Darboux, Sur les développements en série des fonctions d'une seule variable, J. Math. Pures Appl. 3 (2) (1876) 291–312.
[15] D.H. Bailey, K. Jeyabalan, X.S. Li, A comparison of three high-precision quadrature schemes, Experiment. Math. 14 (3) (2005) 317–329.
[16] Wolfram Research, Numerical integration, 2019, https://reference.wolfram.com/language/tutorial/NumericalIntegration.html.
[17] D. Flynn, On the Foundations for a Measure Theory and Integration in Two and Three Dimensions and a Theory of Delta Functions over the Levi–CIvita Field (Master's thesis), University of Manitoba, Department of Physics and Astronomy, 2014.
[18] D. Flynn, K. Shamseddine, On integrable delta functions on the Levi–Civita fields field, P-Adic Numbers Ultrametric Anal. Appl. 10 (1) (2018) 32–56.