

# Exact Real Arithmetic with Automatic Error Estimates in a Computer Algebra System

av

Patrik Andersson

U.U.D.M. Report 2001:P5

Examensarbete i matematik, 20 poäng

Handledare och examinator: Erik Palmgren

1 juni 2001



Department of Mathematics

Uppsala University

# Exact Real Arithmetic with Automatic Error Estimates in a Computer Algebra System

Examensarbete i matemaik  
Patrik Andersson  
Matematiska Institutionen  
Uppsala Universitet

Handledare: Erik Palmgren

1 Juni 2001

## Abstract

The common approach to real arithmetic on computers is floating point arithmetic, which can produce erroneous results due to roundoff errors. An alternative is exact real arithmetic and in this project such arithmetic is implemented in the well-known computer system *Mathematica* by the use of constructive real numbers. All basic operations are implemented as well as the common elementary functions and limits of general convergent sequences of real numbers. Also, as an application to ordinary differential equations, Euler's method for solving initial value problems is implemented.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Preliminaries</b>	<b>2</b>
2.1	The <i>Mathematica</i> Computer Software System . . . . .	2
2.2	Constructive Mathematics . . . . .	4
2.3	Constructive Real Numbers . . . . .	5
2.4	Constructive Real Numbers in <i>Mathematica</i> . . . . .	6
<b>3</b>	<b>The Basic Operations</b>	<b>7</b>
3.1	Comparing Real Numbers . . . . .	7
3.2	Addition and Subtraction . . . . .	8
3.3	Maximum, Minimum and Absolute Value . . . . .	8
3.4	Multiplication . . . . .	9
3.5	Division . . . . .	10
3.6	Sums and Products . . . . .	12
<b>4</b>	<b>The Elementary Functions</b>	<b>14</b>
4.1	Evaluation of Continuous Functions . . . . .	14
4.2	Normalisation . . . . .	15
4.3	The Exponential Function . . . . .	16
4.4	The Trigonometric Functions . . . . .	18
4.5	The Hyperbolic Functions . . . . .	20
4.6	The Logarithm . . . . .	20
4.7	The Inverse Trigonometric Functions . . . . .	22
4.8	The Inverse Hyperbolic Functions . . . . .	24
4.9	Limits of Sequences of Constructive Real Numbers . . . . .	25
4.10	Comparison with Floating Point Arithmetic . . . . .	26
<b>5</b>	<b>Ordinary Differential Equations</b>	<b>27</b>
5.1	Euler's Method and Existence of Solutions . . . . .	27
5.2	Implementation of Euler's Method . . . . .	28
5.3	A Test Example . . . . .	31
<b>A</b>	<b>Source Code</b>	<b>32</b>
<b>B</b>	<b>Test Results</b>	<b>38</b>

# 1 Introduction

The most common approach to computer arithmetic of real numbers is to use floating point approximations, something which can produce erroneous results of certain computations due to roundoff errors. Even relatively simple computations can result in an almost complete loss of accuracy, since only a finite (and from the beginning of the computation fixed) number of decimals can be stored in the memory of the computer (see 3.10 for an example).

An alternative is to use *exact real arithmetic*, which allows exact computations to be performed - exact in the sense that the results are always guaranteed to be correct to a declared precision, and this precision can be set arbitrarily high as long as the memory of the computer is large enough.

In this project, exact real arithmetic is implemented through the use of *constructive real numbers* in the well-known computer mathematics system *Mathematica*.

Section 2 gives a short presentation of *Mathematica* and some background of constructive mathematics/analysis in general and constructive real numbers in particular. Also, the first step of the computer implementation is taken by defining the rational numbers as (embedded into the) constructive reals.

In section 3, the usual basic operations like addition and multiplication are implemented with division being the most interesting. Section 4 contains definitions and implementations of the common elementary functions as well as limits of general convergent sequences of constructive real numbers. Also, a short comparison with floating point arithmetic is made.

The final section shows the possibility of solving ordinary differential equations like initial value problems using an implementation of Euler's method. In the appendices, the complete source code and somewhat extensive test results are presented

## 2 Preliminaries

### 2.1 The *Mathematica* Computer Software System

*Mathematica* is one of the most well-known software systems/computer languages for use in mathematical applications. It is developed and distributed by *Wolfram Research, Inc.*, and is capable of both advanced numerical and symbolic computations. The first version of *Mathematica* was announced in June 1988 for the *Apple Macintosh* and since then versions for all major computer platforms have been released.

In this project all programming and computations were made in *Mathematica* Version 4.1.0.0 running in Linux on a computer having 256MB RAM and an Intel Celeron 375 MHz as CPU.

Most of the *Mathematica* syntax used here is self-explanatory, with a few exceptions which are listed here:

- Arguments of functions are written inside square brackets (and not parentheses). E.g.  $f(2, 3)$  becomes `f[2,3]` in *Mathematica*.
- Variable arguments in function definitions are always followed by an underscore. E.g.  $f(x, y) = x + y$  is written `f[x_,y_] := x+y` in *Mathematica*.
- All built-in functions have names with initial capitals. E.g.  $\sin x$  is written `Sin[x]` in *Mathematica*.
- `If[cond,t,f]` returns `t` if the condition `cond` is true and `f` if the condition is false.
- `Module[{u=val},...commands...]` defines a local variable `u` with initial value `val` and `Module[...]` as scope. (Local variables are in fact used quite often in the following since when it comes to programming, the optimisation abilities of *Mathematica* seem very limited.)
- `{a,b,c,...,z}` defines a list of objects. `Append[L,v]` where `L` is a list appends `v` at the end of the list, e.g. `Append[{a,b,c,...,y},z]` gives the answer `{a,b,c,...,y,z}`. To extract the  $n$ th element from a list `L` is done by the command `L[[n]]` (indexing starts with 1).

Frequently used in the following will be the *Mathematica* notion of a *pure function*. Such a function is defined by `f:=expr&` where `expr` is an expression containing the symbols `#1`, `#2`, ..., `#n`, corresponding to the arguments of  $f$  respectively. For example, addition of three numbers could be defined as `f:=#1+#2+#3&`. The command `f[x,y,z]` would then give the answer  $x+y+z$ , as would the command `f[x][y][z]`.

Also, two of the system parameters of *Mathematica* need to be changed for the computations to work. These are `$IterationLimit`, which gives the maximum length of evaluation chains permitted in the evaluation of any expression, and `$RecursionLimit`, which is the number of levels of recursion permitted.

The default values of these parameters are too low and computations may end prematurely. It is possible to set both parameters to infinity, but this may lead to computer lock-ups due to lack of free memory. Tests have shown

that setting both `$IterationLimit` and `$RecursionLimit` to 500000 will suffice.

More information on the syntax and on *Mathematica* in general is found in [17].

## 2.2 Constructive Mathematics

The fundamental principle of constructive mathematics is that a proof should be a mental/intuitive construction rather than a formal manipulation of formulas and sentences; an idea first presented by the Dutch mathematician Luitzen E. J. Brouwer in 1907. Hence the notion of existence is taken more seriously than in the classical case, and to show constructively that a certain object exists means to give an actual method of constructing it.

In *Intuitionistic Logic*, the logic on which constructive mathematics is founded, to say that a statement  $P$  is true means that there is a proof of  $P$ , while to say that  $P$  is false means that the assumption that  $P$  has a proof leads to a contradiction. A consequence of this notion of truth (and this is the essential difference between intuitionistic logic and classical logic) is that *The Principle of the Excluded Middle*, i.e. for all statements  $P$  necessarily  $P \vee \neg P$ , does not hold.

Even though this fact has far-reaching consequences, large parts of modern analysis have been rebuilt using constructive reasoning only, and the monumental work is [2]. Other works of note are [1] and [4] and these two have been consulted frequently during this project.

A few differences between constructive and classical analysis worth mentioning are:

- For real numbers  $x, y$  it is not always true that  $x = y \vee x \neq y$  (*Principle of Excluded Middle*).
- For real numbers  $x, y$  it is not always true that  $x < y \vee x = y \vee y < x$  (*Law of Trichotomy*).
- There is no constructive proof that continuous functions on compact intervals are uniformly continuous.
- *The Maximum-Minimum Theorem* does not hold constructively.
- *The Intermediate Value Theorem* does not hold constructively, but only an approximate version. Instead of guaranteeing the existence of a point  $x$  for which  $f(x) = 0$ , it can be shown that for every  $\varepsilon > 0$  there exists a point  $x$  with  $|f(x)| < \varepsilon$ .

These facts and much more on the theory of constructive analysis can be found in [1] and [4].

## 2.3 Constructive Real Numbers

A real number is by nature an object containing an infinite amount of information, so to present a real number constructively means to give a rule for calculating it to any desired precision. The definition (which in part coincides with the classical one) is as follows:

**Definition 1** *A real number is a pair  $(x, p)$  consisting of a fundamental sequence of rational numbers  $x : \mathbb{N} \rightarrow \mathbb{Q}$  and a modulus of convergence  $p : \mathbb{N} \rightarrow \mathbb{N}$  such that  $p$  is monotone and for all  $k \in \mathbb{N}$  and all  $m, n \geq p(k)$ ,*

$$|x_m - x_n| \leq 2^{-k}. \quad (1)$$

For simplicity one often writes only  $x$  for the real number, but it is important to remember that the modulus of convergence is always included also when not stated explicitly.

This is but one of several possible definitions of constructive real numbers as  $2^{-k}$  in (1) could be replaced by  $10^{-k}$  or  $\frac{1}{k}$ ,  $k > 0$ . Another common variant (used in [1] and [4]) is  $|x_m - x_n| \leq \frac{1}{m} + \frac{1}{n}$  for  $m, n \in \mathbb{Z}^+$ . The definition stated here is however more convenient for computation purposes and is found in [16].

By taking  $m = p(k)$  and letting  $n \rightarrow \infty$  in (1), it is seen that  $x_{p(k)}$  is an approximation to the real number  $x$  within  $2^{-k}$ . Hence  $x_{p(k)}$  is often called *the  $k$ th approximation of  $x$* .

As there are many different fundamental sequences for the same real number, an equivalence relation for equality has to be defined. Strict inequality is done in a similar way.

**Definition 2** *The real numbers  $x$  and  $y$  are equal, denoted  $x =_{\mathbb{R}} y$ , if for each  $k \in \mathbb{N}$  there is  $m \in \mathbb{N}$  such that for all  $n \geq m$ ,*

$$|x_n - y_n| \leq 2^{-k}.$$

**Definition 3** *For real numbers  $x$  and  $y$ ,  $x <_{\mathbb{R}} y$  if for some  $k \in \mathbb{N}$  there is  $m \in \mathbb{N}$  such that for all  $n \geq m$ ,*

$$y_n - x_n > 2^{-k}.$$

Moreover,  $x \leq_{\mathbb{R}} y$  is defined as  $\neg y <_{\mathbb{R}} x$  and note that this is not equivalent to  $x =_{\mathbb{R}} y \vee x <_{\mathbb{R}} y$  since the law of trichotomy does not hold constructively.

Finally,  $x$  is said to be *apart* from  $y$  if and only if  $x <_{\mathbb{R}} y \vee y <_{\mathbb{R}} x$ , denoted  $x \# y$ . Observe that this is not the same as *not equal to*, since apartness implies the knowledge of either a proof of  $x <_{\mathbb{R}} y$  or a proof of  $y <_{\mathbb{R}} x$  and hence of  $\neg x =_{\mathbb{R}} y$  but that  $\neg x =_{\mathbb{R}} y$  just says that the assumption  $x =_{\mathbb{R}} y$  leads to a contradiction.

**Definition 4** For real numbers  $x$  and  $y$ ,

- (a).  $x \leq_{\mathbb{R}} y \Leftrightarrow_{def} \neg y <_{\mathbb{R}} x$ .
- (b).  $x \# y \Leftrightarrow_{def} x <_{\mathbb{R}} y \vee y <_{\mathbb{R}} x$ .

In the following sections, the subscript  $\mathbb{R}$  will be omitted since it is always clear whether ordinary number relations or relations between constructive real numbers are meant.

## 2.4 Constructive Real Numbers in *Mathematica*

In *Mathematica* the constructive real numbers will have the form  $\mathbf{r}[\mathbf{x}, \mathbf{p}]$  where  $\mathbf{x}$  and  $\mathbf{p}$  are pure functions of one variable corresponding to the fundamental sequence and the modulus of convergence respectively. The "function"  $\mathbf{r}$  is not defined in any particular way, since it is only used to make up the structure of the real numbers.

The only real numbers that can be defined from the start are the rational numbers, and this as real numbers with constant fundamental sequences.

**Definition 5** For every  $a \in \mathbb{Q}$ , the corresponding constructive real number is  $a^* = (a^*, p_{a^*})$  where  $a_n^* = a$  for all  $n \in \mathbb{N}$  and  $p_{a^*}(k) = 0$  for all  $k \in \mathbb{N}$ .

In *Mathematica*:

```
FSconst[a_] := a&
CRconst[a_] := r[FSconst[a], 0&]
```

Here  $\mathbf{a\&}$  is the constant pure function returning  $a$  for any input, and similarly for  $\mathbf{0\&}$ .

Rational numbers could be defined with other fundamental sequences and for example  $Zero = (2^{-n}, k)$  was used for testing purposes. Here the sequence is  $(1, \frac{1}{2}, \frac{1}{4}, \dots)$  and the modulus is  $p(k) = k$  for all  $k \in \mathbb{N}$ , so that  $Zero = 0^*$ .

```
FSzero := 2^(-#1)&
CRzero := r[FSzero, #1&]
```

Frequently used will be the commands to extract the sequence or the modulus from a real number and to compute the approximations of the number.

```
FundSeq[r[x_ , p_]] := x
ConvMod[r[x_ , p_]] := p
Approx[r[x_ , p_], k_] := x[p[k]]
```

The command `Approx[CRzero, k]` would thus give the answer  $2^{-k}$ .

## 3 The Basic Operations

### 3.1 Comparing Real Numbers

To decide by some algorithm that two arbitrary real numbers are equal is not possible, since by definition this means to show that the fundamental sequences of the numbers converge to the same limit and this cannot be done without knowing more about the numbers of those sequences than just the moduli of convergence.

Hence the best one can hope for is to show inequality when it occurs, something which can be achieved using this result:

**Proposition 1** *For any two real numbers  $x = (x, p)$ ,  $y = (y, q)$  it holds that  $x > y \Leftrightarrow x_{p(k)} > y_{q(k)} + 2^{1-k}$  for some  $k \in \mathbb{N}$ .*

Proof: If  $x_{p(k)} > y_{q(k)} + 2^{1-k}$ , then  $x \geq x_{p(k)} - 2^{-k} > y_{q(k)} + 2^{1-k} - 2^{-k} = y_{q(k)} + 2^{-k} \geq y$ , so that  $x > y$ . Conversely, if  $x > y$  then by definition there are  $k, n \in \mathbb{N}$  such that  $x_m > y_m + 2^{-k}$  for all  $m \geq n$ . Hence, with  $m \geq \max(n, p(k+2), q(k+2))$ ,  $x_{p(k+2)} \geq x_m - 2^{-k-2} > y_m + 2^{-k} - 2^{-k-2} \geq y_{q(k+2)} - 2^{-k-2} + 2^{-k} - 2^{-k-2} = y_{q(k+2)} + 2^{-k-1}$ . ■

Thus a function `Compare(x, y)` can be defined that will search for the first natural number  $k$  for which either  $x_{p(k)} > y_{q(k)} + 2^{1-k}$  or  $y_{p(k)} > x_{q(k)} + 2^{1-k}$  and return say  $k + 1$  in the first case and  $-(k + 1)$  in the second so that  $x > y \Leftrightarrow \text{Compare}(x, y) > 0$  and  $y > x \Leftrightarrow \text{Compare}(x, y) < 0$ .

However, if  $x = y$  the search will never stop so a message displayed after some large  $k$  is reached could be useful. In the implementation below, the message *"Possible comparison of equal numbers!"* appears when  $k$  reaches above the global variable `AbortPrecision` set to 10000, and the computation terminates. The value 10000 can of course easily be changed and could even be set to  $\infty$ , thus effectively turning off the message.

```
AbortPrecision:=10000;
```

```

Compare::"equality?"="Possible comparison of equal numbers!";
Compare[r[x_,p_],r[y_,q_],k_]:=
  If[k>AbortPrecision,Message[Compare::"equality?"];Abort[],
    If[x[p[k]]>y[q[k]]+2^(1-k),k+1,
      If[y[q[k]]>y[q[k]]+2^(1-k),-(k+1),
        Compare[r[x,p],r[y,q],k+1]]]]

```

### 3.2 Addition and Subtraction

Addition and subtraction are straightforward to implement using this standard definition:

**Definition 6** For real numbers  $x = (x, p)$  and  $y = (y, q)$ ,

(a).  $x + y := (z, r)$  where  $z_n = x_n + y_n$  and  $r(k) = \max(p(k+1), q(k+1))$  for all  $n, k \in \mathbb{N}$ .

(b).  $-x := (z, r)$  where  $z_n = -x_n$  and  $r(k) = p(k)$  for all  $n, k \in \mathbb{N}$ .

(c).  $x - y := x + (-y)$ .

In case (a) the definition of  $r(k)$  comes from the estimate

$$m, n \geq \max(p(k+1), q(k+1)) \Rightarrow$$

$$|z_m - z_n| = |x_m + y_m - x_n - y_n| \leq |x_m - x_n| + |y_m - y_n| \leq 2^{-(k+1)} + 2^{-(k+1)} = 2^{-k}$$

and in case (b) from the estimate

$$m, n \geq p(k) \Rightarrow |z_m - z_n| = |x_n - x_m| \leq 2^{-k}.$$

In *Mathematica* it looks like this:

```

FSadd[x_,y_] :=x[#1]+y[#1]&
CRadd[r[x_,p_],r[y_,q_]] :=r[FSadd[x,y],Max[p[#1+1],q[#1+1]]&]

```

```

FSneg[x_] :=-x[#1]&
CRneg[r[x_,p_]] :=r[FSneg[x],p]

```

```

CRsub[r[x_,p_],r[y_,q_]] :=CRadd[r[x,p],CRneg[r[y,q]]]

```

### 3.3 Maximum, Minimum and Absolute Value

These operations are also straightforward.

**Definition 7** For real numbers  $x = (x, p)$  and  $y = (y, q)$ ,

(a).  $\max(x, y) := (z, r)$  where  $z_n = \max(x_n, y_n)$  and  $r(k) = \max(p(k), q(k))$  for all  $n, k \in \mathbb{N}$ .

(b).  $\min(x, y) := (z, r)$  where  $z_n = \min(x_n, y_n)$  and  $r(k) = \max(p(k), q(k))$  for all  $n, k \in \mathbb{N}$ .

(c).  $|x| := (z, r)$  where  $z_n = |x_n|$  and  $r(k) = p(k)$  for all  $n, k \in \mathbb{N}$ .

The modulus of convergence for the maximum is derived as follows:

Without loss of generality, suppose that  $x_m = \max(x_m, x_n, y_m, y_n)$ . Then

$$\begin{aligned} m, n \geq \max(p(k), q(k)) &\Rightarrow |z_m - z_n| = |\max(x_m, y_m) - \max(x_n, y_n)| = \\ &= |x_m - \max(x_n, y_n)| = x_m - \max(x_n, y_n) \leq x_m - x_n = |x_m - x_n| \leq 2^{-k}. \end{aligned}$$

The calculation of the modulus for the minimum is similar and as for the absolute value,

$$m, n \geq p(k) \Rightarrow |z_m - z_n| = ||x_m| - |x_n|| \leq |x_m - x_n| \leq 2^{-k}.$$

In *Mathematica*:

```
FSmax[x_, y_] := Max[x[#1], y[#1]] &
CRmax[r[x_, p_], r[y_, q_]] := r[FSmax[x, y], Max[p[#1], q[#1]] &]
```

```
FSmin[x_, y_] := Min[x[#1], y[#1]] &
CRmin[r[x_, p_], r[y_, q_]] := r[FSmin[x, y], Max[p[#1], q[#1]] &]
```

```
FSabs[x_] := Abs[x[#1]] &
CRabs[r[x_, p_]] := r[FSabs[x], p]
```

### 3.4 Multiplication

Again this is straightforward, but the modulus of convergence is a little more complicated to derive.

**Definition 8** For real numbers  $x = (x, p)$  and  $y = (y, q)$ ,  $x \cdot y := (z, r)$  where  $z_n = x_n \cdot y_n$  and

$$r(k) = \max(p(k + 1 + \lceil \log_2(1 + |y_{q(0)}|) \rceil), q(k + 1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil))$$

for all  $n, k \in \mathbb{N}$ .

The derivation of  $r(k)$  is:

$$\begin{aligned}
m, n &\geq \max(p(k+1 + \lceil \log_2(1 + |y_{q(0)}) \rceil), q(k+1 + \lceil \log_2(1 + |x_{p(0)}) \rceil)) \Rightarrow \\
|z_m - z_n| &= |x_m \cdot y_m - x_n \cdot y_n| = |x_m(y_m - y_n) - y_n(x_m - x_n)| \leq \\
&\leq |x_m| \cdot |y_m - y_n| + |y_n| \cdot |x_m - x_n| \leq \\
&\leq |x_m| \cdot 2^{-(k+1 + \lceil \log_2(1 + |x_{p(0)}) \rceil)} + |y_n| \cdot 2^{-(k+1 + \lceil \log_2(1 + |y_{q(0)}) \rceil)} \leq \\
&\leq 2^{-(k+1)} \left( \frac{|x_m|}{1 + |x_{p(0)}|} + \frac{|y_n|}{1 + |y_{q(0)}|} \right) \leq 2^{-(k+1)} (1 + 1) = 2^{-(k+1)} \cdot 2 = 2^{-k}.
\end{aligned}$$

The implementation uses the function  $TwoLog(a)$  which gives the smallest integer  $k \geq 0$  for which  $2^k \geq a$ , i.e.  $\lceil \log_2(a) \rceil$  for  $a \geq 1$ .

```
TwoLog[a_, k_] := If[2^k >= a, k, TwoLog[a, k+1]]
```

```
FSmult[x_, y_] := x[#1] * y[#1] &
CRmult[r[x_, p_], r[y_, q_]] := r[FSmult[x, y],
  Max[p[#1+1+TwoLog[1+Abs[y[q[0]]]], 0]],
  q[#1+1+TwoLog[1+Abs[x[p[0]]]], 0]] &
```

### 3.5 Division

Division is more complicated than the other operations since the inverse of real numbers equal to  $0^*$  is not defined, and even for real numbers apart from  $0^*$  there could be zeros in the fundamental sequence and these have to be replaced or avoided somehow. This is done by padding the sequence, using the following proposition and definition.

**Proposition 2** *For any real number  $x = (x, p)$ ,  $x \neq 0^* \Leftrightarrow |x_{p(k)}| \geq 2^{-(k-1)}$  for some  $k \in \mathbb{N}$ .*

Proof:  $|x_{p(k)}| \geq 2^{-(k-1)} \Rightarrow |x| \geq |x_{p(k)}| - 2^{-k} \geq 2^{-(k-1)} - 2^{-k} = 2^{-k} > 0$  so that  $x \neq 0^*$ . Conversely, if  $x \neq 0^*$  then  $|x| > 0^*$  and hence by definition there are  $k, n \in \mathbb{N}$  such that  $|x_m| > 2^{-k}$  for all  $m \geq n$ , and then, with  $m \geq \max(n, p(k+2))$ ,  $|x_{p(k+2)}| \geq |x_m| - 2^{-(k+2)} \geq 2^{-k} - 2^{-(k+2)} \geq 2^{-(k+1)}$ . ■

**Definition 9** (a). *Let  $x = (x, p)$  be a real number such that  $x \neq 0^*$  and let  $N$  be the smallest natural number such that  $|x_{p(N)}| \geq 2^{-(N-1)}$ . Then  $x^{-1} := (z, r)$  where  $z_n = x_{\max(n, p(N))}^{-1}$  and  $r(k) = p(k + 2N)$  for all  $n, k \in \mathbb{N}$ .*  
(b). *For real numbers  $x, y$  with  $y \neq 0^*$ ,  $\frac{x}{y} := x \cdot y^{-1}$ .*

Note that with this definition  $|z_n| \leq 2^N$  for all  $n \in \mathbb{N}$  and hence

$$\begin{aligned} m, n \geq p(k + 2N) &\Rightarrow |z_m - z_n| = |x_m^{-1} - x_n^{-1}| = |x_m^{-1}| \cdot |x_n^{-1}| \cdot |x_n - x_m| = \\ &= |z_m| \cdot |z_n| \cdot |x_n - x_m| \leq 2^N \cdot 2^N \cdot 2^{-(k+2N)} = 2^{-k}. \end{aligned}$$

In the implementation, the function `CompareToZero(x)` searches for the number  $N$  as defined above (the command is `CompareToZero(|x|)` to be precise). Since the search does not end in case  $x = 0$ , an message similar to the one for `Compare(x, y)` can be useful, and as `CompareToZero` will be used again later when dealing with other functions which are not defined for all real numbers (e.g. the logarithm), the message *"Function possibly not defined (possible division by zero)!"* is appropriate.

`FSpad(x, N)(n) = xmax(n, N)` pads the fundamental sequence and then the inverse is given by

$$\text{CRinv}((x, p(k))) = (1/\text{FSpad}(x, \text{CompareToZero}(|x|)), p(k + 2N)).$$

There is also `CRpad((x, p(k)), N) = (FSpad(x, N), max(p(k), N))` which performs the padding without any other operation and it will be used later.

```

CompareToZero::"undefined?"=
  "Function possibly undefined (possible division by zero)!" ;
CompareToZero [r [x_ , p_ ] , k_] := If [k > AbortPrecision,
  Message [CompareToZero::"undefined?"] ; Abort [],
  If [x [p [k]] >= 2^(1-k) , k, CompareToZero [r [x, p] , k+1]]]

FSpad [x_ , N_] := x [Max [#1, N]] &
CRpad [r [x_ , p_ ] , N_] := r [FSpad [x, N] , Max [p [#1] , N] &]

FSinv [x_ , N_] := 1/FSpad [x, N] [#1] &
CRinv [r [x_ , p_ ] ] := r [FSinv [x, CompareToZero [CRabs [r [x, p]] , 0]] ,
  p [#1+2*CompareToZero [CRabs [r [x, p]] , 0]] &]

CRdiv [r [x_ , p_ ] , r [y_ , q_ ] ] := CRmult [r [x, p] , CRinv [r [y, q]]]

```

As a sidenote, the problem of separating constructive real numbers equal to  $0^*$  from those apart from  $0^*$  could be done by introducing them as different types in some typed  $\lambda$ -calculus. (Errett Bishop [3] was the first to suggest that constructive mathematics can be formulated in typed  $\lambda$ -calculus together with intuitionistic logic.) This would make it possible to avoid the use of a search procedure such as the one above, since the numbers are "marked" as being apart from  $0^*$  or as being equal to  $0^*$ .

### 3.6 Sums and Products

For a general (finite) sum of real numbers  $\sum_{k=1}^n f(k)$ , where  $f : \mathbb{Z}^+ \rightarrow \mathbb{R}$ , if the terms of the sum are added in the order

$$(\dots(((f(1) + f(2)) + f(3)) \dots + f(n-1)) + f(n)),$$

the resulting modulus of convergence for the sum is

$$r(k) = \max(p_1(k+n), p_2(k+n), \dots, p_n(k+n)),$$

where  $p_k$  is the modulus for  $f(k)$ ,  $k = 1, 2, 3, \dots$

Of course the resulting sum is the same whichever order the numbers  $f(k)$  are added, but the modulus of convergence can be improved. If the numbers are added like

$$(\dots(((f(1) + f(2)) + (f(3) + f(4))) + ((f(5) + f(6)) + \dots + f(n)))) \dots),$$

i.e in pairs, the modulus for the sum is

$$r(k) = \max(p_1(k + \lceil \log_2 n \rceil), p_2(k + \lceil \log_2 n \rceil), \dots, p_n(k + \lceil \log_2 n \rceil)),$$

because the number of nested additions has been reduced from  $n$  to  $\lceil \log_2 n \rceil$ .

Since this order of addition is always to be preferred, the previously defined addition function CRadd should perhaps not be used for summing several numbers and a special summation function CRsum( $f, n$ ) could be used instead.

```
MaxSum[f_, n_] :=
  Module[{u=0}, For[k=1, k>=n, u=Max[u, f[k]], k++]; u]

FSsum[f_, n_] := Sum[FundSeq[f[k]] [#1], {k, 1, n}] &
CRsum[f_, n_] :=
  r[FSsum[f, n], MaxSum[ConvMod[f[#1]] & [#1+TwoLog[n, 0]] &, n]]
```

Here MaxSum( $f, n$ ) computes the maximum of  $f(1), \dots, f(n)$ . This is necessary since *Mathematica* does not have a general function for the maximum of a given number of numbers.

For a general product  $\prod_{k=1}^n f(k)$ , such a simple improvement of the modulus of convergence is not possible since the modulus of a product of two numbers is too complicated. However, in the special case when  $f(1) = \dots = f(n) = x$  for some  $x \in \mathbb{R}$  so that  $\prod_{k=1}^n f(k) = x^n$  is an integer power, something can be done. In this case the modulus of convergence for  $x^n$  will be simply

$$p(k + (n-1)(1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil)), \quad (2)$$

where  $p(k)$  is the modulus for  $x$ . This can be proven by induction on  $n$  as follows:

For  $n = 1$  and  $n = 2$ , (2) is obvious, since

$$p(k) = p(k + (1 - 1)(1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil))$$

is the modulus for  $x^1 = x$  and

$$\begin{aligned} \max(p(k + 1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil), p(k + 1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil)) = \\ = p(k + (2 - 1)(1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil)) \end{aligned}$$

is the modulus for  $x^2 = x \cdot x$ , so suppose that (2) holds for some  $n \geq 2$ . Then, the modulus of  $x^{n+1} = x^n \cdot x$  is given by

$$\begin{aligned} \max(p(k + (n - 1)(1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil) + 1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil), \\ p(k + 1 + \lceil \log_2(1 + |(x^n)_{p(0+(n-1)(1+\lceil \log_2(1+|x_{p(0)}|)}|)}|) \rceil)) = \\ = p(\max(k + n(1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil), k + 1 + \lceil \log_2(1 + |(x_{p((n-1)(1+\lceil \log_2(1+|x_{p(0)}|)}|)}|)^n) \rceil)) = \\ = p(k + 1 + \max(n - 1 + n \lceil \log_2(1 + |x_{p(0)}|) \rceil, \lceil \log_2(1 + |x_{p((n-1)(1+\lceil \log_2(1+|x_{p(0)}|)}|)}|)^n) \rceil)) = (*) \end{aligned}$$

and here

$$\begin{aligned} n - 1 + n \lceil \log_2(1 + |x_{p(0)}|) \rceil \geq 1 + n \lceil \log_2(1 + |x_{p(0)}|) \rceil \geq 1 + \lceil n \log_2(1 + |x_{p(0)}|) \rceil = \\ = \lceil 1 + \log_2(1 + |x_{p(0)}|)^n \rceil = \lceil \log_2 2(1 + |x_{p(0)}|)^n \rceil \end{aligned}$$

and

$$\begin{aligned} \lceil \log_2(1 + |x_{p((n-1)(1+\lceil \log_2(1+|x_{p(0)}|)}|)}|^n) \rceil \leq \\ \leq \lceil \log_2(1 + (1 + |x_{p(0)}|)^n) \rceil \leq \lceil \log_2 2(1 + |x_{p(0)}|)^n \rceil \end{aligned}$$

so that

$$(*) = p(k + 1 + (n - 1 + n \lceil \log_2(1 + |x_{p(0)}|) \rceil)) = p(k + n(1 + \lceil \log_2(1 + |x_{p(0)}|) \rceil))$$

which is (2) for  $n + 1$ .

In the implementation below,

$$\text{CRintpow}(x, n) = \begin{cases} x^n & n > 0, x \in \mathbb{R} \\ 1 & n = 0, x \in \mathbb{R} \\ (\frac{1}{x})^{-n} & n < 0, x \in \mathbb{R}, x \neq 0^* \end{cases}$$

```
FSintpow[x_, n_] := (x[#1])^n &
CRintpow[r[x_, p_], n_] := If[n == 0, CRconst[1],
  If[n < 0, CRintpow[CRinv[r[x, p]], -n], r[FSintpow[x, n],
    p[#1 + (n - 1) * (1 + TwoLog[1 + Abs[x[p[0]]], 0]]] &]]]
```

And as a special case deserving its own name,

```
CRsquare[r[x_, p_]] := CRintpow[r[x, p], 2]
```

## 4 The Elementary Functions

### 4.1 Evaluation of Continuous Functions

In constructive analysis, the basic notion of continuity corresponds to the classical notion of uniform continuity and this is because there is no constructive proof that a continuous function on a compact interval is uniformly continuous. Similarly to the modulus of convergence for a real number, a continuous function is equipped with a modulus of continuity.

**Definition 10** *A continuous function is a pair  $(f, v)$  consisting of a function  $f : I \rightarrow \mathbb{R}$ , where  $I$  is a compact interval, and a modulus of continuity  $v : \mathbb{N} \rightarrow \mathbb{N}$  such that  $v$  is monotone and for all  $k \in \mathbb{N}$  and all  $x, y \in I$ ,*

$$|x - y| \leq 2^{-v(k)} \Rightarrow |f(x) - f(y)| \leq 2^{-k}.$$

With the basic operations implemented in the previous section, rational functions such as polynomials with rational coefficients are defined. From this, transcendental functions can be defined as limits of Cauchy sequences of rational functions using the following definition and theorem.

**Definition 11** *A sequence  $\{f_n\}_{n \in \mathbb{N}}$  of continuous functions on a compact interval  $I$  is a Cauchy sequence on  $I$  if there is a monotone function  $u : \mathbb{N} \rightarrow \mathbb{N}$  such that for all  $k \in \mathbb{N}$  and all  $x \in I$ ,*

$$m, n \geq u(k) \Rightarrow |f_m(x) - f_n(x)| \leq 2^{-k}. \quad (3)$$

**Theorem 3** *A sequence  $\{f_n\}_{n \in \mathbb{N}}$  of continuous functions on a compact interval  $I$  converges to a continuous function on  $I$  if and only if it is a Cauchy Sequence.*

Proof: See [4] or [16]. ■

Hence a transcendental continuous function  $f$  on  $I$  can be defined as  $f = (\{f_n\}_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where for each  $n \in \mathbb{N}$ ,  $f_n : I \cap \mathbb{Q} \rightarrow \mathbb{Q}$  is a continuous function and  $v_n : \mathbb{N} \rightarrow \mathbb{N}$  is a modulus of continuity for  $f_n$ . Also,  $u : \mathbb{N} \rightarrow \mathbb{N}$  has the property stated in (3) so that  $\{f_n\}_{n \in \mathbb{N}}$  is a Cauchy sequence of continuous functions. These functions can be for example the partial sums of a Taylor series expansion (as will be the case for the exponential function) or the convergents of a continued fraction (for the logarithm).

Now, given a real number  $x = (x, p)$ , the value of  $f$  at  $x$  is  $f(x) = (y, q)$  where  $y_n = f_n(x_n)$  and  $q(k) = \max(u(k+2), p(v_{u(k+2)}(k+1)))$  for all  $n, k \in \mathbb{N}$ .

That  $f(x)$  is a real number can be seen from the estimate

$$\begin{aligned} m, n \geq \max(u(k+2), p(v_{u(k+2)}(k+1))) &\Rightarrow |f_m(x_m) - f_n(x_n)| \leq \\ &\leq |f_m(x_m) - f_{u(k+2)}(x_m)| + |f_{u(k+2)}(x_m) - f_{u(k+2)}(x_n)| + |f_{u(k+2)}(x_n) - f_n(x_n)| \leq \\ &\leq 2^{-(k+2)} + 2^{-(k+1)} + 2^{-(k+2)} = 2^{-k}. \end{aligned}$$

In *Mathematica* the continuous functions will be of the form `ucf [P,V,U]` where P, V and U are pure functions of two, two and one variable respectively, corresponding to  $f_n(x) = f(n, x)$ ,  $v_n(k) = v(n, k)$  and  $u(k)$  above and from this the implementation of evaluation of continuous functions follows directly:

```
UCFeval [ucf [P_,U_,V_] , r [x_ , p_]] :=
  r [P [#1, x [#1]] &, Max [U [#1+2] , p [V [U [#1+2] , #1+1]]] &]
```

## 4.2 Normalisation

One problem when computing the approximating rational numbers of values of continuous functions, is that the integers in the nominators and the denominators may be very large and this will slow down computations significantly - in particular when approximating values of compositions of functions, e.g.  $x^y = e^{x \cdot \ln y}$ .

To solve this problem, one can apply a kind of "normalising" procedure which will transform a given real number into a "normalised" real number equal to the given one and such that for all approximations the denominator will be of the form  $2^n$  for some  $n \in \mathbb{N}$ .

**Definition 12** *Given a real numbers  $x=(x,p)$ , the normalisation of  $x$  is the real number  $(z,r)$  where  $z_n = \lfloor 2^{n+2} \cdot x_{p(n+2)} \rfloor / 2^{n+2}$  and  $r(k) = k$  for all  $n, k \in \mathbb{N}$ .*

Here  $(z, r) = (x, p)$  as real numbers since for all  $n \in \mathbb{N}$ ,

$$\begin{aligned} 2^{n+2} x_{p(n+2)} &\geq \lfloor 2^{n+2} x_{p(n+2)} \rfloor \geq 2^{n+2} x_{p(n+2)} - 1 \Rightarrow \\ &\Rightarrow x_{p(n+2)} \geq \lfloor 2^{n+2} x_{p(n+2)} \rfloor \cdot 2^{-(n+2)} \geq x_{p(n+2)} - 2^{-(n+2)} \Rightarrow \\ &\Rightarrow |z_n - x_{p(n+2)}| = x_{p(n+2)} - \lfloor 2^{n+2} x_{p(n+2)} \rfloor \cdot 2^{-(n+2)} \leq 2^{-(n+2)}. \end{aligned}$$

Moreover, the definition of the modulus is motivated by the estimate

$$\begin{aligned} m, n \geq k &\Rightarrow |z_m - z_n| \leq |z_m - x_{p(m+2)}| + |x_{p(m+2)} - x_{p(n+2)}| + |x_{p(n+2)} - z_n| \leq \\ &\leq 2^{-(m+2)} + 2^{-(k+2)} + 2^{-(n+2)} \leq 3 \cdot 2^{-(k+2)} \leq 2^{-k}. \end{aligned}$$

In *Mathematica*:

Normalise[r[x\_,p\_]]:=r[Floor[2^(#1+2)\*x[p[#1+2]]]/2^(#1+2)&,#1&]

The simplest way to make use of these normalisations is to do them automatically after each time a continuous function has been evaluated. Hence, the final version of *UCFeval*:

UCFeval[ucf[P\_,V\_,U\_],r[x\_,p\_]]:=  
Normalise[r[P[#1,x[#1]]&,Max[U[#1+2],p[V[U[#1+2],#1+1]]]&]]

### 4.3 The Exponential Function

The exponential function can be defined constructively using the standard Taylor series expansion  $e^x = \lim_{n \rightarrow \infty} \sum_{k=0}^n \frac{x^k}{k!}$ . To do this one must, given a compact interval  $I$ , provide moduli of continuity  $v_n$  for each of the partial sums  $P_n(x) = \sum_{k=0}^n \frac{x^k}{k!}$ ,  $n \in \mathbb{N}$ , together with a function from  $u : \mathbb{N} \rightarrow \mathbb{N}$  showing that the sequence  $\{P_n\}_{n \in \mathbb{N}}$  is a Cauchy sequence.

The *Mathematica* implementation will use only compact intervals of the form  $[-M, M]$  where  $M \in \mathbb{Z}^+$  so it is enough to consider this kind of interval when defining  $\{v_n\}_{n \in \mathbb{N}}$  and  $u$ .

Suppose  $x \in [-M, M]$  for  $M \in \mathbb{Z}^+$  and let  $m, n \in \mathbb{N}$ . Then, with  $t = \min(m, n)$  and  $T = \max(m, n)$ ,

$$\begin{aligned} |P_m(x) - P_n(x)| &= \left| \sum_{k=t+1}^T \frac{x^k}{k!} \right| \leq \sum_{k=t+1}^{\infty} \frac{M^k}{k!} = \frac{M^{t+1}}{(t+1)!} \sum_{k=t+1}^{\infty} \frac{M^{k-t-1}(t+1)!}{k!} \leq \\ &\leq \frac{M^{t+1}}{(t+1)!} \sum_{k=t+1}^{\infty} \frac{M^{k-t-1}}{(k-t-1)!} = \frac{M^{t+1}}{(t+1)!} \sum_{k=0}^{\infty} \frac{M^k}{k!} = \frac{M^{t+1}}{(t+1)!} \cdot e^M \leq \frac{3^M M^{t+1}}{(t+1)!}. \end{aligned}$$

Here note that the last expression is decreasing (in  $t$ ) for  $t+1 \geq M$ . Hence  $u(k)$  can be defined to be the least natural number  $p \geq M$  such that

$$(p+1)! \geq 2^k 3^M M^{p+1}.$$

Now suppose that  $x, y \in [-M, M]$  for  $M \in \mathbb{Z}^+$  and let  $n \in \mathbb{N}$ . Then,

$$\begin{aligned} |P_n(x) - P_n(y)| &= \left| \sum_{k=1}^n \frac{x^k - y^k}{k!} \right| \leq \\ &\leq \sum_{k=1}^n \frac{|x-y| \cdot |x^{k-1} + x^{k-2}y + \dots + xy^{k-2} + y^{k-1}|}{k!} \leq |x-y| \sum_{k=1}^n \frac{k \cdot M^{k-1}}{k!} = \\ &= |x-y| \sum_{k=1}^n \frac{M^{k-1}}{(k-1)!} \leq |x-y| \sum_{k=0}^{\infty} \frac{M^k}{k!} = |x-y| \cdot e^M \leq |x-y| \cdot 3^M, \end{aligned}$$

so that for every  $n \in \mathbb{N}$ ,  $v_n(k)$  can be defined to be the least natural number  $p$  such that

$$3^M 2^{-p} \leq 2^{-k},$$

i.e.  $v_n(k) = k + \lceil \log_2 3^M \rceil$ . This leads to the following definition of the exponential function:

**Definition 13** For  $x \in [-M, M]$  with  $M \in \mathbb{Z}^+$ ,  
 $e^x := (\{P_n(x)\}_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where

$$P_n(x) = \sum_{k=0}^n \frac{x^k}{k!}, v_n(k) = k + \lceil \log_2 3^M \rceil$$

and  $u(k) = \text{least } p \in \mathbb{N} \text{ such that } p \geq M \text{ and } (p+1)! \geq 2^k 3^M M^{p+1}$ .

The translation to *Mathematica* is

```
Pexp[x_, k_, n_] := If[k > n, 1, 1 + (x * Pexp[x, k + 1, n] / k)]
Vexp[n_, k_, M_] := k + TwoLog[3^M, 0]
Uexp[p_, k_, M_] := If[(p + 1)! >= 2^k * 3^M * M^(p + 1), p, Uexp[p + 1, k, M]]
CRexp[x_] := UCFeval[ucf[Pexp[#2, 1, #1] &,
  Vexp[#1, #2, Bound[CRabs[x]]] &,
  Uexp[Bound[CRabs[x]], #1, Bound[CRabs[x]]] &], x]
```

And as a special case, the number  $e = e^{1^*}$ :

```
CRe := CRexp[CRconst[1]]
```

Here  $\text{Bound}(x) : \mathbb{R} \rightarrow \mathbb{Z}^+$  is defined by  $\text{Bound}((x, p)) = \lceil x_{p(1)} + \frac{1}{2} \rceil$ , so that  $x \in [\text{Bound}(x) - 2, \text{Bound}(x)]$ . This can be seen from the following calculation:

$$\begin{aligned} \text{Bound}(x) &= \lceil x_{p(1)} + \frac{1}{2} \rceil \geq x_{p(1)} + \frac{1}{2} \geq x \geq x_{p(1)} - \frac{1}{2} = \\ &= (x_{p(1)} + \frac{1}{2}) - 1 \geq \lceil x_{p(1)} + \frac{1}{2} \rceil - 2 = \text{Bound}(x) - 2. \end{aligned}$$

Thus  $\text{Bound}(|x|)$  provides a positive integer  $M$  for which  $x \in [-M, M]$ .

This method of computing the exponential of a real number turns out to be quite efficient. For example, to compute  $e$  with 1000 decimals (i.e. `Approx[CRe, 3322]` as  $2^{-3322} \approx 10^{-1000}$ ) takes about 0.22 seconds. However, exponentials of real numbers with more complicated fundamental sequences and/or moduli of convergence take more time to evaluate - for example, `Approx[CRexp[CRzero], 333]` computes  $e^{\text{Zero}}$  with 100 decimals and this takes about 0.45 seconds (the answer is of course equal to  $1^*$ ).

It is also useful to note that `Approx[CRexp[CRconst[100]],3322]` takes about 5.49 seconds to compute, while `Approx[CRintpow[CRe,100],3322]` needs only 0.45 seconds. This suggests that the basic operations should be used instead of the continuous function definitions whenever possible.

## 4.4 The Trigonometric Functions

Similarly to the exponential function, the trigonometric functions  $\sin x$  and  $\cos x$  can be defined for  $x \in \mathbb{R}$  by

$$\sin x = \lim_{n \rightarrow \infty} \sum_{k=0}^n \frac{(-1)^k x^{2k+1}}{(2k+1)!}, \cos x = \lim_{n \rightarrow \infty} \sum_{k=0}^n \frac{(-1)^k x^{2k}}{(2k)!}$$

respectively.

Only the calculations of  $u$  and  $v_n$ ,  $n \in \mathbb{N}$ , for  $\sin x$  are given explicitly - the calculations for  $\cos x$  being almost exactly the same. Suppose  $x \in [-M, M]$  for  $M \in \mathbb{Z}^+$  and let  $m, n \in \mathbb{N}$ . Then, with  $t = \min(m, n)$  and  $T = \max(m, n)$ ,

$$\begin{aligned} |P_m(x) - P_n(x)| &= \left| \sum_{k=t+1}^T \frac{(-1)^k x^{2k+1}}{(2k+1)!} \right| \leq \sum_{k=t+1}^{\infty} \frac{M^{2k+1}}{(2k+1)!} = \\ &= \frac{M^{2t+3}}{(2t+3)!} \sum_{k=t+1}^{\infty} \frac{M^{2k-2t-2} (2t+3)!}{(2k+1)!} \leq \frac{M^{2t+3}}{(2t+3)!} \sum_{k=t+1}^{\infty} \frac{M^{2k-2t-2}}{(2k-2t-2)!} = \\ &= \frac{M^{2t+3}}{(2t+3)!} \sum_{k=0}^{\infty} \frac{M^{2k}}{(2k)!} = \frac{M^{2t+3}}{(2t+3)!} \cdot \cosh M = \frac{M^{2t+3}}{(2t+3)!} \frac{e^M + e^{-M}}{2} \leq \\ &\leq \frac{(3^M + 1)M^{2t+3}}{2(2t+3)!}. \end{aligned}$$

The last expression is decreasing for  $2t+3 \geq M$ . Hence,  $u(k)$  can be defined as the least natural number  $p \geq M$  such that  $2(2p+3)! \geq 2^k(3^M + 1)M^{2p+3}$ , for  $k \in \mathbb{N}$ .

Now suppose that  $x, y \in [-M, M]$  for  $M \in \mathbb{Z}^+$  and let  $n \in \mathbb{N}$ . Then,

$$\begin{aligned}
|P_n(x) - P_n(y)| &= \left| \sum_{k=1}^n \frac{(-1)^k (x^{2k+1} - y^{2k+1})}{(2k+1)!} \right| \leq \\
&\leq \sum_{k=1}^n \frac{|x-y| \cdot |x^{2k} + x^{2k-1}y + \dots + xy^{2k-1} + y^{2k}|}{(2k+1)!} \leq \\
&\leq |x-y| \sum_{k=1}^n \frac{(2k+1) \cdot M^{2k}}{(2k+1)!} = |x-y| \sum_{k=1}^n \frac{M^{2k}}{(2k)!} \leq |x-y| \sum_{k=0}^{\infty} \frac{M^{2k}}{(2k)!} = \\
&= |x-y| \cdot \cosh M = |x-y| \frac{e^M + e^{-M}}{2} \leq |x-y| \frac{3^M + 1}{2}.
\end{aligned}$$

Hence, for  $n \in \mathbb{N}$ ,  $v_n(k) = k - 1 + \lceil \log_2(3^M + 1) \rceil$ . This leads to the definition of the sine function, and the definition of the cosine function is obtained in the same way.

**Definition 14** For  $x \in [-M, M]$  with  $M \in \mathbb{Z}^+$ ,

(a).  $\sin x := (\{P_n(x)\}_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where

$$P_n(x) = \sum_{k=0}^n \frac{(-1)^k x^{2k+1}}{(2k+1)!}, v_n(k) = k - 1 + \lceil \log_2(3^M + 1) \rceil$$

and  $u(k) = \text{least } p \in \mathbb{N} \text{ such that } p \geq M \text{ and } 2(2p+3)! \geq 2^k(3^M + 1)M^{2p+3}$ ,

(b).  $\cos x := (\{P_n(x)\}_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where

$$P_n(x) = \sum_{k=0}^n \frac{(-1)^k x^{2k}}{(2k)!}, v_n(k) = k - 1 + \lceil \log_2 3^M \rceil$$

and  $u(k) = \text{least } p \in \mathbb{N} \text{ such that } p \geq M \text{ and } 2(2p+2)! \geq 2^k(3^M + 1)M^{2p+2}$ .

The *Mathematica* implementation can be found in Appendix A.

To compute  $\sin 1^*$  with 1000 decimals takes about 0.15 seconds and the same for  $\cos 1^*$ , which is somewhat faster than  $e^{1^*}$  with the same accuracy, as would be expected from the formulas above.

The other trigonometric functions are conveniently defined as derived from  $\sin$  and  $\cos$ . For example,  $\tan x := \sin x / \cos x$  (for  $\cos x \neq 0^*$ ) and  $\cot x := \cos x / \sin x$  (for  $\sin x \neq 0^*$ ). Using the first definition,  $\tan 1^*$  with 1000 decimals takes 0.33 seconds to compute.

## 4.5 The Hyperbolic Functions

The Hyperbolic functions  $\sinh$ ,  $\cosh$ ,  $\tanh$  and  $\coth$  are implemented in exactly the same way as the corresponding trigonometric functions. Hence the definitions are given directly:

**Definition 15** For  $x \in [-M, M]$  with  $M \in \mathbb{Z}^+$ ,

(a).  $\sinh x := (\{P_n(x)\}_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where

$$P_n(x) = \sum_{k=0}^n \frac{x^{2k+1}}{(2k+1)!}, v_n(k) = k - 1 + \lceil \log_2(3^M + 1) \rceil$$

and  $u(k) = \text{least } p \in \mathbb{N} \text{ such that } p \geq M \text{ and } 2(2p+3)! \geq 2^k(3^M + 1)M^{2p+3}$ ,

(b).  $\cosh x := (\{P_n(x)\}_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where

$$P_n(x) = \sum_{k=0}^n \frac{x^{2k}}{(2k)!}, v_n(k) = k - 1 + \lceil \log_2 3^M \rceil$$

and  $u(k) = \text{least } p \in \mathbb{N} \text{ such that } p \geq M \text{ and } 2(2p+2)! \geq 2^k(3^M + 1)M^{2p+2}$ .

Moreover,  $\tanh x := \sinh x / \cosh x$  for all  $x \in \mathbb{R}$  and  $\coth x := \cosh x / \sinh x$  for  $x \neq 0^*$ .

The computation times are almost exactly the same as for the trigonometric functions in all cases (see appendix B).

## 4.6 The Logarithm

To define the natural logarithm, a continued fraction expansion will be implemented instead of the Taylor series, as the former has the major advantage of converging for all  $x \in \mathbb{R}$  for which the logarithm is defined, i.e. for all  $x > 0^*$ , and this much faster than the Taylor series. The continued fraction used is this ([6], [11]):

$$\ln(x) = (x-1) \left( \frac{a_0}{x-1} + \frac{x-1}{1} + \frac{a_1}{x-1} + \frac{x-1}{1} + \frac{a_2}{x-1} + \frac{x-1}{1} + \frac{a_3}{x-1} + \frac{x-1}{1} + \dots \right)$$

for  $x > 0$ , where  $a_0 = a_{2k+1} = 1$  for  $n \geq 0$  and  $a_{2n} = \frac{n}{n+1}$  for  $n \geq 1$ .

A convenient estimate exists for the difference between the convergents of this continued fraction. For  $n \in \mathbb{N}$ , let  $P_n(x)$  be the  $n$ th convergent, i.e.

$$P_n(x) = (x-1) \left( \frac{a_0}{x-1} + \frac{x-1}{1} + \frac{a_1}{x-1} + \frac{x-1}{1} + \dots + \frac{a_n}{x-1} + \frac{x-1}{1} \right).$$

Then, for  $x > 0$  and  $m, n \in \mathbb{N}$  and with  $t = \min(m, n)$ ,

$$|P_m(x) - P_n(x)| \leq \frac{(x-1)^2}{x} \left| \frac{1-\sqrt{x}}{1+\sqrt{x}} \right|^{t-1}.$$

This function is decreasing for  $x < 1$  and increasing for  $x > 1$ , so to use this estimate with a constructive real number  $x$ , both upper and lower bounds for the value of  $x$  is needed. The upper bound is  $M = \text{Bound}(x)$  as done previously, and the lower bound is  $2^{-N}$  where  $N = \text{CompareToZero}(x)$  so that  $x \in [2^{-N}, M]$  is asserted. Thus  $u(k)$  can be defined as the least natural number  $p$  such that

$$\max \left( \frac{(M-1)^2}{M} \left( \frac{\sqrt{M}-1}{\sqrt{M}+1} \right)^{p-1}, \frac{(1-2^{-N})^2}{2^{-N}} \left( \frac{1-\sqrt{2^{-N}}}{1+\sqrt{2^{-N}}} \right)^{p-1} \right) \leq 2^{-k}.$$

Moreover, to give an estimate for the difference  $|P_n(x) - P_n(y)|$  for  $x, y \in [2^{-N}, M]$  it is enough to note that for all  $n \in \mathbb{N}$ ,

$$\left| \frac{d}{dx} P_n(x) \right| \leq \left| \frac{1}{x} \right| \leq \frac{1}{2^{-N}} = 2^N,$$

so that  $|P_n(x) - P_n(y)| \leq |x - y| \cdot 2^N$  and  $v_n(k) = k + N$ .

As a final twist, it is obviously necessary to apply the logarithm to padded real numbers  $\text{CRpad}(x, \text{CompareToZero}(x))$ , since otherwise the sequence of  $P_n$ 's might be calculated for non-positive rational numbers and hence not converge. This is also where the "check" is made to see that the function really is defined for the input number  $x$ .

Now the results above may be collected in the definition used for the *Mathematica* implementation of the logarithm:

**Definition 16** For  $x \in [2^{-N}, M]$  with  $M \in \mathbb{Z}^+$  and  $N \in \mathbb{N}$ ,  
 $\ln x := (\{P_n(\text{CRpad}(x, \text{CompareToZero}(x)))\}_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where

$$P_n(x) = (x-1) \left( \frac{a_0}{x} - \frac{x-1}{1} + \frac{a_1}{x} - \frac{x-1}{1} + \cdots + \frac{a_n}{x} - \frac{x-1}{1} \right),$$

with  $a_0 = a_{2k+1} = 1$  for  $n \geq 0$ ,  $a_{2n} = \frac{n}{n+1}$  for  $n \geq 1$ ,  $v_n(k) = k + N$  and  $u(k) = \text{least } p \in \mathbb{N} \text{ such that}$

$$\max \left( \frac{(M-1)^2}{M} \left( \frac{\sqrt{M}-1}{\sqrt{M}+1} \right)^{p-1}, \frac{(1-2^{-N})^2}{2^{-N}} \left( \frac{1-\sqrt{2^{-N}}}{1+\sqrt{2^{-N}}} \right)^{p-1} \right) \leq 2^{-k}.$$

To compute  $\ln 2^*$  and  $\ln 10^*$  with 1000 decimals takes about 2.1 seconds and 8.6 seconds respectively, which is far better than in case the Taylor series would have been used.

From the natural logarithm it is now simple to derive other elementary functions, such as

- general powers,  $x^y := e^{y \cdot \ln x}$  for  $x > 0^*$ ,  $y \in \mathbb{R}$ ,
- the square root,  $\sqrt{x} := x^{(\frac{1}{2})^*}$  for  $x > 0^*$ ,
- general logarithms,  $\log_a x := \frac{\ln x}{\ln a}$  for  $a, x > 0^*$ .

As an example, the square root of two with 100 decimals require a computation time of about 1.7 seconds. More extensive test results are presented in Appendix B.

## 4.7 The Inverse Trigonometric Functions

In this section, arctan and arcsin will be defined as continued fractions similarly to the logarithm. Other inverse trigonometric functions can then be derived from these two.

A suitable continued fraction expansion for  $\arctan x$ ,  $x \in \mathbb{R}$  is ([5])

$$\arctan x = x \left( \frac{1}{1} + \frac{x^2}{3} + \frac{4x^2}{5} + \frac{9x^2}{7} + \dots \right).$$

Here the continued fraction inside the parentheses has partial quotients with positive nominators and denominators and hence the standard continued fraction estimate of convergence can be used ([11]), i.e. for  $m, n \in \mathbb{N}$  with  $t = \min(m, n)$  and

$$\frac{A_n(x)}{B_n(x)} = \frac{1}{1} + \frac{x^2}{3} + \frac{4x^2}{5} + \frac{9x^2}{7} + \dots + \frac{n^2 x^2}{2n+1},$$

it holds that

$$\left| \frac{A_m(x)}{B_m(x)} - \frac{A_n(x)}{B_n(x)} \right| \leq \left| \frac{A_t(x)}{B_t(x)} - \frac{A_{t-1}(x)}{B_{t-1}(x)} \right|$$

and here  $A_n(x)$  and  $B_n(x)$  can be computed by the recursions

$$A_{-1}(x) = 0, A_0 = 1, A_n = (2n+1)A_{n-1}(x) + n^2 x^2 A_{n-2}(x) \text{ for } n \geq 2,$$

$$B_{-1}(x) = 1, B_0 = 1, B_n = (2n+1)B_{n-1}(x) + n^2 x^2 B_{n-2}(x) \text{ for } n \geq 2.$$

Thus, if  $P_n(x) = x \frac{A_n(x)}{B_n(x)}$  for  $n \in \mathbb{N}$  and  $x \in [-M, M]$  for  $M \in \mathbb{Z}^+$ , then

$$\begin{aligned} |P_m(x) - P_n(x)| &= \left| x \frac{A_m(x)}{B_m(x)} - x \frac{A_n(x)}{B_n(x)} \right| = |x| \left| \frac{A_m(x)}{B_m(x)} - \frac{A_n(x)}{B_n(x)} \right| \leq \\ &\leq M \left| \frac{A_t(x)}{B_t(x)} - \frac{A_{t-1}(x)}{B_{t-1}(x)} \right| \leq M \left| \frac{A_t(M)}{B_t(M)} - \frac{A_{t-1}(M)}{B_{t-1}(M)} \right|. \end{aligned}$$

This suggests that  $u(k)$  for arctan can be defined as the least natural number such that the last expression in the formula above is less than or equal to  $2^{-k}$ . To define the functions  $v_n(k)$  is simple, since for all  $n \in \mathbb{N}$  and all  $x \in \mathbb{R}$ ,

$$\left| \frac{d}{dx} P_n(x) \right| \leq 1$$

so that  $|P_n(x) - P_n(y)| \leq |x - y|$  for  $x, y \in \mathbb{R}$  and  $v_n(k) = k$ .

**Definition 17** For  $x \in [-M, M]$  with  $M \in \mathbb{Z}^+$ ,  
arctan  $x := (\{P_n(x)\}_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where

$$P_n(x) = x \left( \frac{1}{1} + \frac{x^2}{3} + \frac{4x^2}{5} + \frac{9x^2}{7} + \cdots + \frac{n^2 x^2}{2n+1} \right),$$

$v_n(k) = k$  and  $u(k) = \text{least } p \in \mathbb{N} \text{ such that}$

$$M \left| \frac{A_p(M)}{B_p(M)} - \frac{A_{p-1}(M)}{B_{p-1}(M)} \right| \leq 2^{-k},$$

where  $A_n(x), B_n(x)$  are defined by the recursion formulas above.

This implementation of arctan in *Mathematica* is reasonably fast - for example the number  $\pi$  defined as  $\pi := 4^* \text{arctan } 1^*$  with 1000 decimals is done in 5.6 seconds.

For arcsin  $x$ ,  $-1^* < x < 1^*$ , a slightly more complicated continued fraction expansion will be used, namely ([5])

$$\frac{\arcsin x}{\sqrt{1-x^2}} = x \left( \frac{1}{1-x^2} + \frac{x^2}{3} + \frac{4x^2}{5(1-x^2)} + \frac{9x^2}{7} + \frac{16x^2}{9(1-x^2)} + \frac{25x^2}{11} + \cdots \right).$$

Once again, the continued fraction inside the parentheses have partial quotients with positive nominators and denominators, so the same kind of estimate as for arctan can be used to define  $u(k)$  here as well. The recursion formulas for  $A_n(x)$  and  $B_n(x)$  in this case are

$$\begin{aligned} A_{-1}(x) &= 0, A_0 = 1, \\ A_{2n+1} &= (4n+3)A_{2n}(x) + (2n+1)^2 x^2 A_{2n-1}(x) \text{ for } n \geq 0, \\ A_{2n} &= (4n+1)(1-x^2)A_{2n-1}(x) + 4n^2 x^2 A_{2n-2}(x) \text{ for } n \geq 1, \end{aligned}$$

$$\begin{aligned}
B_{-1}(x) &= 1, B_0 = 1, \\
B_{2n+1} &= (4n+3)B_{2n}(x) + (2n+1)^2x^2B_{2n-1}(x) \text{ for } n \geq 0, \\
B_{2n} &= (4n+1)(1-x^2)B_{2n-1}(x) + 4n^2x^2B_{2n-2}(x) \text{ for } n \geq 1.
\end{aligned}$$

Let  $N = \text{CompareToZero}(1^* - |x|)$ , so that  $x \in [-1+2^{-N}, 1-2^{-N}]$  is assured. Then, with  $P_n(x) = x \frac{A_n(x)}{B_n(x)}$  and  $n \in \mathbb{N}$ ,

$$\left| \frac{d}{dx} P_n(x) \right| \leq \left| \frac{d}{dx} \frac{x}{1-x^2} \right| = \frac{1+x^2}{(1-x^2)^2} \leq \frac{1+(1-2^{-N})^2}{(1-(1-2^{-N})^2)^2} \leq 2^{2N}$$

so that  $v_n(k) = k + 2N$ .

**Definition 18** For  $x \in [-1+2^{-N}, 1-2^{-N}]$  with  $N \in \mathbb{N}$ ,  $\arcsin x := \sqrt{1-x^2}(P_n(\text{CRpad}(x, \text{CompareToZero}(1^* - |x|))))_{n \in \mathbb{N}}, \{v_n\}_{n \in \mathbb{N}}, u)$  where

$$\begin{aligned}
P_{2n}(x) &= x \left( \frac{1}{1-x^2} + \frac{x^2}{3} + \frac{4x^2}{5(1-x^2)} + \frac{9x^2}{7} + \cdots + \frac{4n^2x^2}{(4n+1)(1-x^2)} \right), \\
P_{2n+1}(x) &= x \left( \frac{1}{1-x^2} + \frac{x^2}{3} + \frac{4x^2}{5(1-x^2)} + \frac{9x^2}{7} + \cdots + \frac{(2n+1)^2x^2}{4n+3} \right),
\end{aligned}$$

$v_n(k) = k + 2N$  and  $u(k) = \text{least } p \in \mathbb{N} \text{ such that}$

$$\left| \frac{A_p(1-2^{-N})}{B_p(1-2^{-N})} - \frac{A_{p-1}(1-2^{-N})}{B_{p-1}(1-2^{-N})} \right| \leq 2^{-k},$$

where  $A_n(x), B_n(x)$  are defined by the recursion formulas above.

Since the implementation of  $\arcsin$  involves the evaluation of a square root, it is considerably slower than that for  $\arctan$  (see Appendix B).

From this the other inverse trigonometric functions can be derived, e.g.  $\arccos x := \frac{\pi}{2} - \arcsin x$  for  $-1^* < x < 1^*$  and  $\text{arccot } x := \arctan x^{-1}$  for  $x \neq 0^*$ .

## 4.8 The Inverse Hyperbolic Functions

The inverse hyperbolic functions are implemented as their respective logarithmic representations, i.e.

**Definition 19** (a).  $\text{arsinh } x := \ln(x + \sqrt{x^2 + 1^*})$  for all  $x \in \mathbb{R}$ ,

(b).  $\text{arcosh } x := \ln(x + \sqrt{x^2 - 1^*})$  for  $x > 1^*$ ,

(c).  $\text{artanh } x := \left(\frac{1}{2}\right)^* \ln \frac{1^*+x}{1^*-x}$  for  $-1^* < x < 1^*$ ,

(d).  $\text{arcoth } x := \left(\frac{1}{2}\right)^* \ln \frac{x+1^*}{x-1^*}$  for  $x < -1^*$  or  $x > 1^*$ .

Naturally, the computations of  $\text{arsinh}$  and  $\text{arcosh}$  will require more time than those of  $\text{artanh}$  and  $\text{arcoth}$  since the former two functions involves a square root while the latter two do not. See Appendix B for details.

## 4.9 Limits of Sequences of Constructive Real Numbers

The final notion to be implemented in this section is limits of explicitly defined sequences of real numbers. Obviously, as in the case of sequences of continuous functions, sequences of real numbers converge if and only if they are Cauchy sequences.

**Definition 20** *If  $\{b_m = (b_m, p_m)\}_{m \in \mathbb{N}}$  is a Cauchy sequence of real numbers and  $c : \mathbb{N} \rightarrow \mathbb{N}$  is monotone and such that  $m, n \geq c(k) \Rightarrow |b_m - b_n| \leq 2^{-k}$  for all  $k \in \mathbb{N}$ , then the limit of the sequence is  $\lim_{m \rightarrow \infty} (b_m, c) := (z, r)$  where  $z_n = (b_{c(n)})_{p_{c(n)}(n)}$  and  $r(k) = k + 2$  for all  $n, k \in \mathbb{N}$ .*

Thus the  $n$ th term in the fundamental sequence of the limiting real number is term number  $p_{c(n)}(n)$  in the fundamental sequence of the  $c(n)$ th number in the Cauchy sequence. The calculation of the modulus of convergence for the limit is

$$\begin{aligned} m, n \geq k + 2 &\Rightarrow |z_m - z_n| = |(b_{c(m)})_{p_{c(m)}(m)} - (b_{c(n)})_{p_{c(n)}(n)}| \leq \\ &\leq |(b_{c(m)})_{p_{c(m)}(m)} - b_{c(m)}| + |b_{c(m)} - b_{c(n)}| + |b_{c(n)} - (b_{c(n)})_{p_{c(n)}(n)}| \leq \\ &\leq 2^{-m} + 2^{-(k+2)} + 2^{-n} \leq 2^{-(k+2)} + 2^{-(k+2)} + 2^{-(k+2)} \leq 2^{-k} \end{aligned}$$

and in *Mathematica* it looks quite simple:

```
CRlimit[B_,C_] :=
  r[Module[{m=B[C[#1]]},FundSeq[m][ConvMod[m][#1]]]&,#1+2&]
```

For example, a number that can be defined by such a sequence is Euler's constant  $\gamma$ . An estimate of convergence is: ([7])

$$0 < \gamma - \left( \sum_{k=1}^m \frac{1}{k} - \frac{1}{2m} + \frac{1}{12m^2} - \frac{1}{120m^4} - \ln m \right) \leq \frac{1}{252m^6}$$

Hence  $\gamma := \lim_{m \rightarrow \infty} (b_m, c)$ , where

$$b_m = \left( \sum_{k=1}^m \frac{1}{k} - \frac{1}{2m} + \frac{1}{12m^2} - \frac{1}{120m^4} \right)^* - \ln m,$$

and  $c(k) = \text{least } p \in \mathbb{N} \text{ such that } 252p^6 \geq 2^k$ . In *Mathematica*:

```
Bgamma:=CRsub[CRconst[HarmonicNumber[#1]-
  1/(2*#1)+1/(12*(#1)^2)-1/(120*(#1)^4)],CRln[CRconst[#1]]]&
Cgamma[p_,k_] :=If[252*p^6>=2^k,p,Cgamma[p+1,k]]
CRgamma:=CRlimit[Bgamma,Cgamma[1,#1]&]
```

## 4.10 Comparison with Floating Point Arithmetic

In most cases the exact real arithmetic based on constructive real numbers implemented here is considerably slower than the standard floating point arithmetic used normally by *Mathematica*.

The continuous functions defined by their Taylor series, like the exponential function and the trigonometric functions, fare quite well in comparison while the others, like the logarithm, do not. Even worse in comparison are compositions of functions like the square root, since the computations become more complex with an increasing number of nested function calls and increased demand for accuracy at each step.

However, it is in the last point above that the main advantage of exact real arithmetic lies; consider this computation, taken from [10]:

```
x:=10^(-100)
N[(1-cos[x])/x^2,100]
```

Here `N[... ,100]` means that *Mathematica* should compute the expression, i.e.  $\frac{1-\cos x}{x^2}$  for  $x = 10^{-100}$ , with a precision of 100 digits. But since the program does not know the amount of extra precision which will be needed from the start, it will in fact be unable to do the calculation and will report an error:

```
In increasing internal precision while attempting to evaluate, the
limit $MaxExtraPrecision = 50 was reached. Increasing the value of
$MaxExtraPrecision may help resolve the uncertainty.
```

Certainly an increase in the mentioned system parameter `$MaxExtraPrecision` would resolve the problem, but one does not know with how much it should be increased and for other similar expressions the computations may still fail (e.g. 1000 instead of 100 digits accuracy).

With exact real arithmetic the extra precision needed is estimated automatically at each step so that the computation goes through without any problems. The calculation of the 333rd approximation (which gives 100 decimals of accuracy) of the expression above is done by the command

```
Approx[CRmult[CRsub[CRconst[1],CRCos[CRconst[10^(-100)]]],
CRconst[10^200]],333]
```

The calculation takes about 3.3 seconds and the answer is (as it should be) a rational numbers close to, but not exactly,  $\frac{1}{2}$ .

## 5 Ordinary Differential Equations

### 5.1 Euler's Method and Existence of Solutions

Given numbers  $a, b, s \in \mathbb{R}$  with  $a < b$  and a continuous function  $f : [a, b] \times \mathbb{R} \rightarrow \mathbb{R}$ , consider the *initial value problem*

$$\begin{cases} x(a) = s \\ x'(t) = f(t, x(t)) \quad a \leq t \leq b. \end{cases} \quad (4)$$

The standard theorem on existence and uniqueness of solutions to such a problem is Picard's theorem:

**Theorem 4** *Suppose  $f$  satisfies a Lipschitz condition in  $x$  for all values in a neighbourhood of the point  $(a, s)$  in  $\mathbb{R} \times \mathbb{R}$ . Then there is an open interval around  $a$  where the initial value problem (4) has a unique solution.*

That  $f$  satisfies a *Lipschitz condition* in  $x$  means that there is a *Lipschitz constant*  $L \geq 0$  such that for all  $(t, x), (t, x')$  in the mentioned neighbourhood,

$$|f(t, x) - f(t, x')| \leq L|x - x'|.$$

As is noted in [1], the original proof given by Picard (found for example in [8]) using successive approximations (a.k.a. Picard's method) is constructively valid.

Similar to the successive approximations method is the well-known method of Euler, for which the *polygonal* (i.e. piecewise linear) *approximations* of the solution to (4) is constructed as follows. For  $n \in \mathbb{N}$ , let the *step size* be  $h = (b - a)/2^n$  and let the *lattice points* on  $[a, b]$  be  $t_i = a + ih$  for  $i = 0, 1, \dots, 2^n$ . Then the  $n$ th polygonal approximation is the function  $u_n : \mathbb{R} \rightarrow \mathbb{R}$  defined by

$$\begin{cases} u_n(t_0) = s \\ u_n(t_i) = u_n(t_{i-1}) + hf(t_i, u_n(t_{i-1})) \quad 1 \leq i \leq 2^n. \end{cases} \quad (5)$$

A constructive proof of the convergence of Euler's method can be found in [15] (i.e. convergence to a solution of (4), so that this also works as a constructive proof of the existence of such a solution) and it is here shown that the sequence of polygonal approximations  $\{u_n\}_{n \in \mathbb{N}}$  is a Cauchy sequence. The estimate of convergence is that for each  $k \in \mathbb{N}$  if  $m, n \geq k$ , then

$$|u_m(t) - u_n(t)| \leq \frac{e^{(b-a)L} - 1}{L}(\omega(h) + hLY),$$

for all  $t \in [a, b]$ , where

- $L > 0$  is a Lipschitz constant for  $f$ ,
- $h = \frac{b-a}{2^k}$ ,
- $Y = e^{(b-a)L}|s| + \frac{e^{(b-a)L}-1}{L}C$  so that  $|u_n(t)| \leq Y$  for all  $n \in \mathbb{N}$ ,  $t \in [a, b]$ ,
- $C = \sup\{|f(t, 0)| : t \in [a, b]\}$ ,
- $\omega : \mathbb{R}^+ \rightarrow \mathbb{R}$  is a modulus of continuity for  $f$ , i.e. for  $\varepsilon > 0$ ,  $\omega(\varepsilon) = \sup\{|f(t, x) - f(t', x)| : |t - t'| \leq \varepsilon \text{ and } t, t' \in [a, b], x \in [-Y, Y]\}$ .

Given an initial value problem with the necessary information specified above (i.e.  $f$ ,  $a$ ,  $b$ ,  $s$ ,  $L$ ,  $C$  and  $\omega$ ) and a point in  $[a, b]$  where the solution should be calculated, this estimate can be used to implement Euler's method in *Mathematica* as a Cauchy sequence of constructive real numbers. This will be done next.

## 5.2 Implementation of Euler's Method

To simplify matters, the implementation of Euler's method will be restricted to use only rational numbers as the endpoints of the interval ( $a$  and  $b$ ) and the initial value ( $s$ ).

The first step is to implement the sequence of approximating functions  $\{u_n\}_{n \in \mathbb{N}}$  and this will be done by using a general polygonal function as defined in [15]:

**Definition 21** *Given a finite sequence of pairs of real numbers  $\{(a_k, b_k)\}_{k=0}^n$  with  $a_0 < a_1 < \dots < a_n$ , the polygonal function  $\mathcal{P} : \mathbb{R} \rightarrow \mathbb{R}$  is defined as*

$$\mathcal{P}_{(a_k, b_k)}(x) := b_0 + \sum_{k=0}^{n-1} \frac{b_{k+1} - b_k}{a_{k+1} - a_k} (\max(0, x - a_k) - \max(0, x - a_{k+1})).$$

**Proposition 5**  $\mathcal{P} : \mathbb{R} \rightarrow \mathbb{R}$  is a uniformly continuous function and

- $\mathcal{P}_{(a_k, b_k)}(x) = b_0$  for  $x \leq a_0$ ,
- $\mathcal{P}_{(a_k, b_k)}(x) = \frac{x - a_{k+1}}{a_k - a_{k+1}}(b_k - b_{k+1}) + b_{k+1}$  for  $a_k \leq x \leq a_{k+1}$ ,  $0 \leq k \leq n-1$ ,
- $\mathcal{P}_{(a_k, b_k)}(x) = b_n$  for  $x \geq a_n$ .

Proof: In [15].

Now, given the initial value problem (4) and a fixed  $n \in \mathbb{N}$ , let  $h = (b-a)/2^n$  and put  $t_k = a + kh$  for  $k \geq 0$ ,  $a_0 = s$  and  $a_k = a_{k-1} + hf(t_{k-1}, a_{k-1})$  for  $k \geq 1$ . Then clearly  $\mathcal{P}_{(t_k, a_k)}(t) = u_n(t)$  for all  $t \in [a, b]$  i.e. the polygonal function

determined by the two sequences equals the  $n$ th polygonial approximation according to Euler's method defined in (5).

The *Mathematica* version of the Polygonial function takes as input two lists of length  $2^n + 1$  of rational numbers, corresponding to the sequences of points above. Here note that in the sequence corresponding to the  $a_k$ 's, the  $n$ th approximation of the continuous constructive reals function corresponding to  $f$  is used in each step. This will introduce a "roundoff" error so that the estimate of convergence for the sequence of  $u_n$ 's given earlier have to be modified somewhat.

```
Apoints[a_, b_, n_] := Module[{u={a}, h=(b-a)*2^(-n), k=1},
  For[k=1, k<=2^n-1, u=Append[u, a+h*k], k++] ; u]
```

```
Bpoints[f_, a_, b_, s_, n_] :=
  Module[{v={s}, h=(b-a)*2^(-n), l=1, A=Apoints[a, b, n]},
    For[l=1, l<=2^n-1, v=Append[v, v[[l-1]]+h*
      Approx[Normalise[f[CRconst[A[[l-1]]]],
        CRconst[v[[l-1]]]], n]], l++] ; v]
```

```
Polygonial[f_, a_, b_, s_, n_, x_] :=
  Module[{A=Apoints[a, b, n], B=Bpoints[f, a, b, s, n]},
    B[[1]]+Sum[((B[[k+1]]-B[[k]])/(A[[k+1]]-A[[k]]))*
      (Max[0, x-A[[k]]]-Max[0, x-A[[k+1]]]), {k, 1, 2^n-1}]]
```

An initial value problem in *mathematica* will look like `ivp[f, a, b, s, L, C, m]` with notations as before ( $m$  corresponding to  $\omega$ ) and the approximations  $u_n$  will be the terms of a Cauchy sequence, each taking such an initial value problem together with a rational number  $x$  as arguments.

```
Beuler[ivp[f_, a_, b_, s_, L_, C_, m_], x_] :=
  CRconst[Polygonial[f, a, b, s, #1, x]] &
```

It remains to give an estimate of convergence for this sequence. As is stated in [8], the local roundoff error (i.e. the round-off error in a single step) in Euler's method is the sum of the error induced by rounding the product  $hf^*$ , where  $f^*$  is the approximation of  $f$ , and the error induced by the inaccuracy of the evaluation of  $f$ , i.e.  $h|f - f^*|$ . In the implementation used here, both  $h$  and  $f^*$  are rational numbers so that no error appears when rounding  $hf^*$ , and since  $f^*$  is nothing else than the  $n$ th approximation of  $f$ , the error induced when evaluating  $f$  is at most  $h2^{-n}$ .

By a theorem in [8], if the local roundoff error is at most  $\varepsilon$  at each step, then the accumulated roundoff error  $r$  after any number of steps can be

estimated by

$$|r| \leq \frac{\varepsilon}{h} E_L(b-a),$$

where  $E_L : \mathbb{R} \rightarrow \mathbb{R}$  is the *Lipschitz function* defined by

$$E_L(x) = \begin{cases} x & L = 0 \\ \frac{e^x - 1}{L} & L > 0 \end{cases}$$

and  $L \geq 0$  is a Lipschitz constant for  $f$ . From above it follows directly that  $|r| \leq 2^{-n} E_L(b-a)$ . The Lipschitz function in *Mathematica* is

```
Lipschitz[a_, b_, L_] := If [L==0, b-a, (3^(L*(b-a))-1)/L]
```

where  $e$  is changed to 3 to speed up computations. Now the total error for the implementation can be estimated from the above result and the estimate in the previous subsection (incorporating the case  $L = 0$  as in [8]). For each  $k \in \mathbb{N}$  (with  $h = (b-a)/2^k$ ),

$$\begin{aligned} m, n \geq k &\Rightarrow |u_m - u_n| \leq \\ &\leq E_L(b-a) \cdot (\omega(h) + hL(e^{(b-a)L}|s| + E_L(b-a) \cdot C)) + 2^{-k} E_L(b-a) = \\ &= 2^{-k} E_L(b-a) \cdot (2^k \omega(\frac{b-a}{2^k}) + (b-a)L(e^{(b-a)L}|s| + E_L(b-a) \cdot C) + 1). \end{aligned}$$

This leads to the definition of a function  $c : \mathbb{N} \rightarrow \mathbb{N}$  showing that the sequence of implemented polygonal approximations is Cauchy as follows.

$$\begin{aligned} c(k) = \text{least } p \in \mathbb{N} \text{ such that } p \geq k + \lceil \log_2(\text{Lipschitz}(a, b, L) \cdot \\ \cdot (2^p \omega(\frac{b-a}{2^p}) + (b-a)L(3^{(b-a)L}|s| + \text{Lipschitz}(a, b, L) \cdot C) + 1)) \rceil \end{aligned}$$

In *Mathematica* it looks like this:

```
Ceuler[ivp[f_, a_, b_, s_, L_, C_, m_], p_, k_] :=
  If [p>=k+TwoLog[Lipschitz[a,b,L]*
    (m[(b-a)/2^p]*2^p+(b-a)*L*(3^(L*(b-a))*Abs[s]+
    Lipschitz[a,b,L]*C)+1), 0], p,
    Euler[ivp[f, a, b, s, L, C, m], p+1, k]]
```

Finally, the value of the solution to the initial value problem at a point  $x$  is given by the limit of the sequence.

```
IVPeuler[ivp[f_, a_, b_, s_, L_, C_, m_], x_] :=
  CRlimit[Beuler[ivp[f, a, b, s, L, C, m], x],
    Euler[ivp[f, a, b, s, L, C, m], 1, #1]&]
```

### 5.3 A Test Example

Unfortunately, the presented implementation of Euler's method turns out to be surprisingly slow. Even for very simple initial value problems (i.e. for simple functions  $f(t, x)$ ) only a few approximations can be computed in a reasonably short time. An example is the problem

$$\begin{cases} x(0) = 1 \\ x'(t) = x + t \quad 0 \leq t \leq 1. \end{cases}$$

The data necessary for solving this problem is  $a = 0$ ,  $b = 1$ ,  $s = 1$ ,  $L = 1$ ,  $C = 1$  and  $\omega(\varepsilon) = \varepsilon$  so to compute the  $n$ th approximation to the solution in the point  $x = 1$  is done by the following command:

```
Approx[IVPeuler[ivp[CRadd[#1,#2]&,0,1,1,1,1,#1&],1],n]
```

However, already for  $n = 6$  the calculation takes more than 17 seconds (the result is 3.43518... — quite close to the analytic solution  $2e - 2 = 3.43656\dots$ ). The probable explanation for this is that number of nested function calls grows very (even extremely) large with higher accuracies and perhaps these types of calculations is not something *Mathematica* is optimised to perform. The evaluation is done from the inside and out, so to speak.

More test results can be found in Appendix B.

## A Source Code

```
Off[General::spell]
Off[General::spell1]

$IterationLimit=500000;
$RecursionLimit=500000;

FSconst[a_]:=a&
CRconst[a_]:=r[FSconst[a],0&]

FSzero:=2^(-#1)&
CRzero:=r[FSzero,#1&]

FundSeq[r[x_,p_]]:=x
ConvMod[r[x_,p_]]:=p
Approx[r[x_,p_],k_]:=x[p[k]]

AbortPrecision:=10000;

Compare::"equality?"="Possible comparison of equal numbers!";
Compare[r[x_,p_],r[y_,q_],k_]:=
  If[k>AbortPrecision,Message[Compare::"equality?"];Abort[],
    If[x[p[k]]>y[q[k]]+2^(1-k),k+1,
      If[y[q[k]]>y[q[k]]+2^(1-k),-(k+1),
        Compare[r[x,p],r[y,q],k+1]]]]]

FSadd[x_,y_]:=x[#1]+y[#1]&
CRadd[r[x_,p_],r[y_,q_]]:=r[FSadd[x,y],Max[p[#1+1],q[#1+1]]&]

FSneg[x_]:=-x[#1]&
CRneg[r[x_,p_]]:=r[FSneg[x],p]

CRsub[r[x_,p_],r[y_,q_]]:=CRadd[r[x,p],CRneg[r[y,q]]]

FSmax[x_,y_]:=Max[x[#1],y[#1]]&
CRmax[r[x_,p_],r[y_,q_]]:=r[FSmax[x,y],Max[p[#1],q[#1]]&]

FSmin[x_,y_]:=Min[x[#1],y[#1]]&
CRmin[r[x_,p_],r[y_,q_]]:=r[FSmin[x,y],Max[p[#1],q[#1]]&]
```

```

FSabs[x_] := Abs[x[#1]] &
CRabs[r[x_, p_]] := r[FSabs[x], p]

TwoLog[a_, k_] := If[2^k >= a, k, TwoLog[a, k+1]]

FSmult[x_, y_] := x[#1] * y[#1] &
CRMult[r[x_, p_], r[y_, q_]] := r[FSmult[x, y],
  Max[p[#1+1+TwoLog[1+Abs[y[q[0]]]], 0]],
  q[#1+1+TwoLog[1+Abs[x[p[0]]]], 0]] &

CompareToZero := "undefined?" =
  "Function possibly undefined (possible division by zero)!" ;
CompareToZero[r[x_, p_], k_] :=
  If[k > AbortPrecision, Message[CompareToZero := "undefined?"];
  Abort[], If[x[p[k]] >= 2^(1-k), k, CompareToZero[r[x, p], k+1]]

FSpad[x_, N_] := x[Max[#1, N]] &
CRpad[r[x_, p_], N_] := r[FSpad[x, N], Max[p[#1], N]] &

FSinv[x_, N_] := 1/FSpad[x, N][#1] &
CRinv[r[x_, p_]] := r[FSinv[x, CompareToZero[CRabs[r[x, p]], 0]],
  p[#1+2*CompareToZero[CRabs[r[x, p]], 0]] &

CRdiv[r[x_, p_], r[y_, q_]] := CRMult[r[x, p], CRinv[r[y, q]]]

CRsign[r[x_, p_]] := CRdiv[r[x, p], CRabs[r[x, p]]]

MaxSum[f_, n_] :=
  Module[{u=0}, For[k=1, k>=n, u=Max[u, f[k]], k++]; u]

FSSum[f_, n_] := Sum[FundSeq[f[k]][#1], {k, 1, n}] &
CRsum[f_, n_] :=
  r[FSSum[f, n], MaxSum[ConvMod[f[#1]] & [#1+TwoLog[n, 0]] &, n]]

FSintpow[x_, n_] := (x[#1])^n &
CRintpow[r[x_, p_], n_] := If[n==0, CRconst[1],
  If[n<0, CRintpow[CRinv[r[x, p]], -n], r[FSintpow[x, n],
  p[#1+(n-1)*(1+TwoLog[1+Abs[x[p[0]]]], 0)]] &]]]

CRsquare[r[x_, p_]] := CRintpow[r[x, p], 2]

```

```

Normalise[r[x_,p_]]:=r[Floor[2^(#1+2)*x[p[#1+2]]]/2^(#1+2)&,#1&]

UCFeval[ucf[P_,V_,U_],r[x_,p_]]:=
  Normalise[r[P[#1,x[#1]]&,Max[U[#1+2],p[V[U[#1+2],#1+1]]]&]]

Bound[r[x_,p_]]:=Ceiling[x[p[1]]+1/2]

Pexp[x_,k_,n_]:=If[k>n,1,1+(x*Pexp[x,k+1,n]/k)]
Vexp[n_,k_,M_]:=k+TwoLog[3^M,0]
Uexp[p_,k_,M_]:=If[(p+1)!>=2^k*3^M*M^(p+1),p,Uexp[p+1,k,M]]
CRexp[x_]:=UCFeval[ucf[Pexp[#2,1,#1]]&,
  Vexp[#1,#2,Bound[CRabs[x]]]&,
  Uexp[Bound[CRabs[x]],#1,Bound[CRabs[x]]]&],x]

CRe:=CRexp[CRconst[1]]

Psin[x_,k_,n_]:=If[k>n,
  x,x+(-1)*x^2*Psin[x,k+1,n]/((2*k)*(2*k+1))]
Vsin[n_,k_,M_]:=k-1+TwoLog[3^M+1,0]
Usin[p_,k_,M_]:=If[2*(2*p+3)!>=2^k*M^(2*p+3)*(3^M+1),
  p,Usin[p+1,k,M]]
CRsin[x_]:=UCFeval[ucf[Psin[#2,1,#1]]&,
  Vsin[#1,#2,Bound[CRabs[x]]]&,
  Usin[Bound[CRabs[x]],#1,Bound[CRabs[x]]]&],x]

Pcos[x_,k_,n_]:=If[k>n,
  1,1+(-1)*x^2*Pcos[x,k+1,n]/((2*k-1)*(2*k))]
Vcos[n_,k_,M_]:=k-1+TwoLog[3^M,0]
Ucos[p_,k_,M_]:=If[2*(2*p+2)!>=2^k*M^(2*p+2)*(3^M+1),
  p,Ucos[p+1,k,M]]
CRCos[x_]:=UCFeval[ucf[Pcos[#2,1,#1]]&,
  Vcos[#1,#2,Bound[CRabs[x]]]&,
  Ucos[Bound[CRabs[x]],#1,Bound[CRabs[x]]]&],x]

CRtan[x_]:=CRdiv[CRsin[x],CRCos[x]]

CRcot[x_]:=CRdiv[CRCos[x],CRsin[x]]

Psinh[x_,k_,n_]:=If[k>n,
  x,x+x^2*Psinh[x,k+1,n]/((2*k)*(2*k+1))]
Vsinh[n_,k_,M_]:=k-1+TwoLog[3^M+1,0]

```

```

Usinh[p_,k_,M_] := If [2*(2*p+3)! >= 2^k*M^(2*p+3)*(3^M+1),
  p, Usinh[p+1,k,M]]
CRsinh[x_] := UCFeval [ucf [Psinh[#2,1,#1]&,
  Vsinh[#1,#2,Bound[CRabs[x]]]&,
  Usinh[Bound[CRabs[x]],#1,Bound[CRabs[x]]]&],x]

Pcosh[x_,k_,n_] := If [k>n,
  1, 1+x^2*Pcosh[x,k+1,n]/((2*k-1)*(2*k))]
Vcosh[n_,k_,M_] := k-1+TwoLog[3^M,0]
Ucosh[p_,k_,M_] := If [2*(2*p+3)! >= 2^k*M^(2*p+3)*(3^M+1),
  p, Ucosh[p+1,k,M]]
CRcosh[x_] := UCFeval [ucf [Pcosh[#2,1,#1]&,
  Vcosh[#1,#2,Bound[CRabs[x]]]&,
  Ucosh[Bound[CRabs[x]],#1,Bound[CRabs[x]]]&],x]

CRtanh[x_] := CRdiv[CRsinh[x],CRcosh[x]]

CRcoth[x_] := CRdiv[CRcosh[x],CRsing[x]]

Pln[x_,k_,n_] :=
  If [k==0 | Mod[k,2]==1, 1, k/(k+2)]/(1+x-x/(1+Pln[x,k+1,n]))]
Vln[n_,k_,N_] := k+N
Uln[p_,k_,M_,N_] :=
  If [(M-1)^2*2^k*(Sqrt[M]-1)^(p-1) <= M*(Sqrt[M]+1)^(p-1)
    && (1-2^(-N))^2*2^(k+N)*(1-Sqrt[2^(-N)])^(p-1)
    <= (1+Sqrt[2^(-N)])^(p-1), p, Uln[p+1,k,M,N]]
CRln[x_] := UCFeval [ucf [(FundSeq[x][#1]-1)*Pln[#2,0,#1]&,
  Vln[#1,#2,CompareToZero[x,0]]&,
  Uln[2,#1,Bound[x],CompareToZero[x,0]]&],
  CRsub[CRpad[x,ConvMod[x]
  [CompareToZero[x,0]]],CRconst[1]]]

CRpow[x_,y_] := CRexp[CRmult[y,CRln[x]]]

CRsqrt[x_] := CRpow[x,CRconst[1/2]]

CRlog[a_,x_] := CRdiv[CRln[x],CRln[a]]

Parctan[x_,k_,n_] := If [k>n,
  0, If [k==0, 1, (k*x)^2]/(2*k+1+Parctan[x,k+1,n])]
Varctan[n_,k_] := k

```

```

Uarctan[p_,k_,M_,A_,B_,C_,D_] :=
  If[p-1<k || M*2^k*Abs[C*B-A*D]>Abs[B*D],
    Uarctan[p+1,k,M,C,D,(2*p+1)*C+(p*M)^2*A,
      (2*p+1)*D+(p*M)^2*B],p]
CRarctan[x_] :=UCFeval[ucf[FundSeq[x][#1]*Parctan[#2,0,#1]&,
  Varctan[#1,#2]&,Uarctan[0,#1,Bound[CRabs[x]],0,1,1,1]&],x]

CRpi:=CRmult[CRconst[4],CRarctan[CRconst[1]]]

Parcsin[x_,k_,n_] :=If[k>n,If[k==0,1/(1-x^2+Parcsin[x,k+1,n]),
  If[Mod[k,2]==1,(k*x)^2/(2*k+1+Parcsin[x,k+1,n]),
    (k*x)^2/((2*k+1)*(1-x^2)+Parcsin[x,k+1,n])]]]
Varcsin[n_,k_,N_] :=k+2*N
Uarcsin[p_,k_,N_,A_,B_,C_,D_] :=
  If[p-1<k || 2^k*Abs[C*B-A*D]>Abs[B*D],
    Uarcsin[p+1,k,N,C,D,(2*k+1)*If[Mod[k,2]==1,
      1,2^(1-N)-2^(-2*N)]*C+(k*(1-2^(-N)))^2*A,(2*k+1)*
      If[Mod[k,2]==1,
        1,2^(1-N)-2^(-2*N)]*D+(k*(1-2^(-N)))^2*B],p-1]
CRarcsin[x_] :=
  Module[{u=CompareToZero[CRsub[CRconst[1],CRabs[x]],0]},
    CRmult[CRsqrt[CRsub[CRconst[1],CRsquare[x]]],
      UCFeval[ucf[FundSeq[x][#1]*Parcsin[#2,0,#1]&,
        Varcsin[#1,#2,u]&,Uarcsin[0,#1,u,0,1,1,1]&],
        CRpad[x,ConvMod[x][u]]]]]

CRarccos[x_] :=
  CRsub[CRmult[CRconst[2],CRarctan[CRconst[1]]],CRarcsin[x]]

CRarccot[x_] :=CRarctan[CRinv[x]]

CRarcsinh[x_] :=
  CRln[CRadd[x,CRsqrt[CRadd[CRsquare[x],CRconst[1]]]]]

CRarccosh[x_] :=
  CRln[CRadd[x,CRsqrt[CRsub[CRsquare[x],CRconst[1]]]]]

CRarctanh[x_] :=CRmult[CRconst[1/2],
  CRln[CRdiv[CRadd[CRconst[1],x],CRsub[CRconst[1],x]]]]

CRarccoth[x_] :=CRmult[CRconst[1/2],

```

```

CRln[CRdiv[CRadd[x,CRconst[1]],CRsub[x,CRconst[1]]]]]

CRlimit[B_,C_]:=
  r[Module[{m=B[C[#1]]},FundSeq[m][ConvMod[m][#1]]&,#1+2&]

Bgamma:=CRsub[CRconst[HarmonicNumber[#1]-
  1/(2*#1)+1/(12*(#1)^2)-1/(120*(#1)^4)],CRln[CRconst[#1]]]&
Cgamma[p_,k_]:=If[252*p^6>=2^k,p,Cgamma[p+1,k]]
CRgamma:=CRlimit[Bgamma,Cgamma[1,#1]&]

Apoints[a_,b_,n_]:=Module[{u={a},h=(b-a)*2^(-n),k=1},
  For[k=1,k<=2^n-1,u=Append[u,a+h*k],k++];u]

Bpoints[f_,a_,b_,s_,n_]:=
  Module[{v={s},h=(b-a)*2^(-n),l=1,A=Apoints[a,b,n]},
  For[l=1,l<=2^n-1,v=Append[v,v[[l-1]]+h*
    Approx[Normalise[f[CRconst[A[[l-1]]],
    CRconst[v[[l-1]]]]],n]],l++];v]

Polygonial[f_,a_,b_,s_,n_,x_]:=
  Module[{A=Apoints[a,b,n],B=Bpoints[f,a,b,s,n]},
  B[[1]]+Sum[[(B[[k+1]]-B[[k]])/(A[[k+1]]-A[[k]])*
    (Max[0,x-A[[k]]]-Max[0,x-A[[k+1]]])],{k,1,2^n-1}]]

Beuler[ivp[f_,a_,b_,s_,L_,C_,m_],x_]:=
  CRconst[Polygonial[f,a,b,s,#1,x]]&

Lipschitz[a_,b_,L_]:=If[L==0,b-a,(3^(L*(b-a))-1)/L]

Ceuler[ivp[f_,a_,b_,s_,L_,C_,m_],p_,k_]:=
  If[p>=k+TwoLog[Lipschitz[a,b,L]*
    (m[(b-a)/2^p]*2^p+(b-a)*L*(3^(L*(b-a))*Abs[s]+
    Lipschitz[a,b,L]*C)+1),0],p,
  Ceuler[ivp[f,a,b,s,L,C,m],p+1,k]]

IVPeuler[ivp[f_,a_,b_,s_,L_,C_,m_],x_]:=
  CRlimit[Beuler[ivp[f,a,b,s,L,C,m],x],
  Ceuler[ivp[f,a,b,s,L,C,m],1,#1]&]

```

## B Test Results

Function	Precision (decimals)	Result	Time (seconds)
CR <sub>e</sub>	1000	$e$	0.22
CR <sub>exp</sub> [CR <sub>const</sub> [100]]	1000	$e^{100}$	5.49
CR <sub>intpow</sub> [CR <sub>e</sub> , 100]	1000	$e^{100}$	0.44
CR <sub>intpow</sub> [CR <sub>e</sub> , 1000]	1000	$e^{1000}$	6.01
CR <sub>exp</sub> [CR <sub>zero</sub> ]	100	$e^0 = 1$	0.45
CR <sub>sin</sub> [CR <sub>const</sub> [1]]	1000	$\sin(1)$	0.15
CR <sub>cos</sub> [CR <sub>const</sub> [1]]	1000	$\cos(1)$	0.15
CR <sub>tan</sub> [CR <sub>const</sub> [1]]	1000	$\tan(1)$	0.33
CR <sub>cot</sub> [CR <sub>const</sub> [1]]	1000	$\cot(1)$	0.33
CR <sub>sinh</sub> [CR <sub>const</sub> [1]]	1000	$\sinh(1)$	0.15
CR <sub>cosh</sub> [CR <sub>const</sub> [1]]	1000	$\cosh(1)$	0.15
CR <sub>tanh</sub> [CR <sub>const</sub> [1]]	1000	$\tanh(1)$	0.32
CR <sub>coth</sub> [CR <sub>const</sub> [1]]	1000	$\coth(1)$	0.32
CR <sub>ln</sub> [CR <sub>const</sub> [2]]	1000	$\ln(2)$	2.12
CR <sub>ln</sub> [CR <sub>const</sub> [10]]	1000	$\ln(10)$	8.59
CR <sub>ln</sub> [CR <sub>const</sub> [1/2]]	1000	$\ln(\frac{1}{2})$	4.12
CR <sub>sqrt</sub> [CR <sub>const</sub> [2]]	100	$\sqrt{2}$	1.64
CR <sub>sqrt</sub> [CR <sub>const</sub> [10]]	100	$\sqrt{10}$	2.08
CR <sub>pow</sub> [CR <sub>const</sub> [2], CR <sub>const</sub> [1/3]]	100	$\sqrt[3]{2}$	1.74
CR <sub>log</sub> [CR <sub>const</sub> [10], CR <sub>const</sub> [2]]	100	$\lg(2)$	0.69
CR <sub>pi</sub>	1000	$\pi$	5.61
CR <sub>arcsin</sub> [CR <sub>const</sub> [1/2]]	100	$\arcsin(\frac{1}{2})$	2.87
CR <sub>arccos</sub> [CR <sub>const</sub> [1/2]]	100	$\arccos(\frac{1}{2})$	3.06
CR <sub>arccot</sub> [CR <sub>const</sub> [1]]	1000	$\operatorname{arccot}(1)$	5.59
CR <sub>arcsinh</sub> [CR <sub>const</sub> [1/2]]	50	$\operatorname{arcsinh}(\frac{1}{2})$	2.83
CR <sub>arccosh</sub> [CR <sub>const</sub> [2]]	50	$\operatorname{arccosh}(2)$	2.39
CR <sub>arctanh</sub> [CR <sub>const</sub> [1/2]]	1000	$\operatorname{arctanh}(\frac{1}{2})$	2.44
CR <sub>arccoth</sub> [CR <sub>const</sub> [2]]	1000	$\operatorname{arccoth}(2)$	2.44
CR <sub>gamma</sub>	20	$\gamma$	1.65
CR <sub>sqrt</sub> [CR <sub>pi</sub> ]	50	$\sqrt{\pi}$	2.45
CR <sub>ln</sub> [CR <sub>pi</sub> ]	50	$\ln(\pi)$	1.26
CR <sub>exp</sub> [CR <sub>pi</sub> ]	100	$e^\pi$	1.73
CR <sub>pow</sub> [CR <sub>pi</sub> , CR <sub>e</sub> ]	50	$\pi^e$	2.82
CR <sub>pow</sub> [CR <sub>sqrt</sub> [CR <sub>const</sub> [2]], CR <sub>sqrt</sub> [CR <sub>const</sub> [2]]]	20	$\sqrt{2}^{\sqrt{2}}$	9.74
CR <sub>exp</sub> [CR <sub>neg</sub> [CR <sub>gamma</sub> ]]	20	$e^{-\gamma}$	2.12
CR <sub>exp</sub> [CR <sub>mult</sub> [CR <sub>sqrt</sub> [CR <sub>const</sub> [163]], CR <sub>pi</sub> ]]	10	$e^{\sqrt{163}\pi}$	17.02

Initial value problem 1: 
$$\begin{cases} x(0) = 0 \\ x'(t) = 2t \quad 0 \leq t \leq 1. \end{cases}$$

$a = 0, b = 1, s = 0, L = 0, C = 2, \omega(\varepsilon) = 2\varepsilon.$

Analytic solution:  $x = t^2$ . Value at  $b$ :  $x(1) = 1$ .

Initial value problem 2: 
$$\begin{cases} x(0) = 1 \\ x'(t) = x \quad 0 \leq t \leq 1. \end{cases}$$

$a = 0, b = 1, s = 1, L = 1, C = 0, \omega(\varepsilon) = 0.$

Analytic solution:  $x = e^t$ . Value at  $b$ :  $x(1) = e = 2.71828\dots$

Initial value problem 3: 
$$\begin{cases} x(0) = 1 \\ x'(t) = x + t \quad 0 \leq t \leq 1. \end{cases}$$

$a = 0, b = 1, s = 1, L = 1, C = 1, \omega(\varepsilon) = \varepsilon.$

Analytic solution:  $x = 2e^t - t - 1$ . Value at  $b$ :  $x(1) = 2e - 2 = 3.43656\dots$

Initial value problem 4: 
$$\begin{cases} x(0) = 1 \\ x'(t) = tx \quad 0 \leq t \leq 1. \end{cases}$$

$a = 0, b = 1, s = 1, L = 1, C = 0, \omega(\varepsilon) = 2\varepsilon.$

Analytic solution:  $x = e^{\frac{t^2}{2}}$ . Value at  $b$ :  $x(1) = e^{\frac{1}{2}} = 1.64872\dots$

Initial value problem no.	Approximation no.	Value (at $x = b$ )	Time (seconds)
1	3	0.99206...	0.18
1	5	0.99803...	0.81
1	8	0.99975...	19.16
2	3	2.70161...	0.23
2	5	2.71410...	1.33
2	7	2.71723...	16.92
3	3	3.42556...	0.61
3	5	3.43380...	4.44
3	6	3.43518...	17.89
4	3	1.64625...	0.81
4	5	1.64810...	5.38
4	6	1.64841...	19.36

## References

- [1] Michael J. Beeson. *Foundations of Constructive Mathematics*. Springer-Verlag, Berlin-Heidelberg-New York-Tokyo, 1985.
- [2] Errett Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, New York, 1967.
- [3] Errett Bishop. Mathematics as a numerical language. *Intuitionism and Proof Theory*, pages 53–71, 1970.
- [4] Errett Bishop and Douglas Bridges. *Constructive Analysis*, volume 279 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, Berlin-Heidelberg-New York-Tokyo, 1985.
- [5] Aleksej Nikolaevic Chovanskij. *The Application of Continued Fractions and their Generalizations to Problems in Approximation Theory*. Library of Applied Analysis and Computational Mathematics. P. Noordhoff N.V., Groningen, The Netherlands, 1963.
- [6] William B. Gragg. Truncation error bounds for  $g$ -fractions. *Numerische Mathematik*, 11:370–379, 1968.
- [7] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, Inc., 2nd edition, 1994.
- [8] Peter Henrici. *Discrete Variable Methods in Ordinary Differential Equations*. John Wiley and Sons Inc., New York-London, 1962.
- [9] Peter Henrici. *Elements of Numerical Analysis*. John Wiley and Sons Inc., New York-London-Sydney, 1964.
- [10] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM Publishing, Philadelphia, 1996.
- [11] William B. Jones and Wolfgang J. Thron. *Continued Fractions: Analytic Theory and Applications*, volume 11 of *Encyclopedia of Mathematics and its applications*. Cambridge University Press, 1984.
- [12] Alexander M. Ostrowski. *Solution of Equations and Systems of Equations*. Academic Press Inc., New York and London, 2nd edition, 1966.
- [13] Erik Palmgrem. Constructive real numbers in mathematica. Unpublished Manuscript, 2000.

- [14] Erik Palmgren. A constructive approach to nonstandard analysis. *Annals of Pure and Applied Logic*, 73:297–325, 1995.
- [15] Erik Palmgren. Constructive nonstandard analysis. In A. Petry, editor, *Méthodes et analyse non standard*, volume 9 of *Cahiers du Centre de Logique*, pages 69–97, 1996.
- [16] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics: An introduction, vol. 1*. North-Holland Publ. Co., Amsterdam, 1988.
- [17] Stephen Wolfram. *Mathematica - A system for Doing Mathematics by Computer*. Addison-Wesley Publishing Company Inc., 2nd edition, 1991.