Accurate Computation of Chaotic Dynamical Systems

WALTER KRÄMER
University of Wuppertal
Faculty of Mathematics and Natural Sciences
42 119 Wuppertal
GERMANY

Abstract: The computation of orbits of *dynamical systems* is known to be highly unstable if the system exhibits chaotic behavior. In this case, even for the very simplest systems, ordinary floating-point computations will eventually deliver results which are completely wrong quantitatively, when compared with the true trajectory on which the computation began. Similarly, ordinary interval arithmetic (i. e. intervals of floating-point numbers) yield poor enclosures after few iterations. In most cases the computation breaks down because of overflow. Using `intpakX`'s multiple precision intervals, we can compute enclosures of orbits for a considerably longer time with high accuracy. Statements concerning the sensitivity with respect to small changes in the seed value of the numerical computations are possible with mathematical rigor. We also show that computing an orbit using a rational arithmetic as e.g. provided by the computer algebra system Maple is not possible due to computing time and computer memory limitations.

Key–Words: Dynamical system, logistic equation, chaotic behaviour, verified numerical results, `intpakX`

1 Introduction

Let us cite from [13]:

> The logistic map is a polynomial mapping, often cited as an archetypal example of how complex, chaotic behaviour can arise from very simple non-linear dynamical equations. The map was popularized in a seminal 1976 paper by the biologist Robert May. The logistic model was originally introduced as a demographic model by Pierre Franois Verhulst. Later it was applied on surplus production of the population biomass of species in the presence of limiting factors such as food supply or disease, and so two causal effects:
>
> - reproduction means the population will increase at a rate proportional to the current population
>
> - starvation means the population will decrease at a rate proportional to the value obtained by taking the theoretical "carrying capacity" of the environment less the current population.
>
> Mathematically this can be written as
>
> $$x_{n+1} = ax_n(1 - x_n)$$
>
> where x_n is a number between zero and one, and represents the population at year n, and hence x_0 represents the initial population (at year 0). a is a positive number, and represents a combined rate for reproduction and starvation.

Most values of a beyond 3.57 up to 4.0 exhibit chaotic behaviour. The system exhibits dynamics that are very sensitive to the initial condition (here the value of a). The accurate numerical computation of the iterates x_n for larger $n \in \mathbb{N}$ is a nontrivial task. In the following we will first give an example showing the difficulties in computing x_n using Maple's decimal arithmetic(s) for different values of mantissa digits k. We will also demonstrate that ordinary interval computations will break down very soon due to overestimations. Computations with rational numbers (starting from rationals x_0 and a) are only possible for small values of n. The numerators and denominators become very soon very large. The computing time is increasing rapidly and the amount of computer memory needed is a limiting factor. To overcome these problems we will use a multiple precision interval arithmetic based on Maples multiple precision decimal arithmetic. We will compare the naive interval computation (linear approximation property) with a so called centered form (quadratic approximation property). The multiple precision interval arithmetic has been implemented as part of a software pack-

age called `intpakX` providing different kinds of interval algorithms (range enclosures, interval Newton method, complex interval operations, ...). The package is freely available as a Maple power tool. A link to the newest version of `intpakX` is given in Section 3.

2 Accurate numeical computation of orbits of a chaotic dynamical system

The computation of an orbit of a dynamic system is known to be highly unstable if the system exhibits chaotic behavior. In this case, even for the very simplest systems, ordinary floating-point computations will eventually deliver results which are completely wrong quantitatively, when compared with the true trajectory on which the computation began. Similarly, ordinary interval arithmetic (i. e. intervals of floating-point numbers) yields poor enclosures after few iterations. In most cases the computation breaks down because of overflow. Using `intpakX`'s multiple precision intervals, we can compute enclosures of orbits for a considerably longer time with high accuracy.

Consider the simple dynamic system as given by the *logistic equation*:

$$x_{n+1} = a \cdot x_n \cdot (1 - x_n)\,, \;\; n \;\geq\; 0 \qquad (1)$$

for some $a \in [0, 4]$ and $x_0 \in (0, 1)$.

On the computer, we can compute this iteration with

(i) ordinary floating-point arithmetic,

(ii) ordinary interval arithmetic, or

(iii) multiple precision interval arithmetic.

However, for the cases (ii) and (iii) it would be better to first rewrite the right hand side of (1) such that it is better suited for the application of interval arithmetic: For narrow intervals it is well known in interval analysis that a tighter interval enclosure can be obtained by using a *mean value form* instead of an interval evaluation of the originally given expression.

The ordinary interval evaluation of a function $f(x)$ over an interval X, denoted as $f(X)$, is obtained via replacing all occurrences of x in f by the interval X and via replacing all operations by the corresponding interval operations. The mean value form is defined by $f_m(X) := f(y) + f'(X)(X - y)$ with some fixed value $y \in X$, e.g., the midpoint. Thus, in the cases (ii) and (iii) we may replace the right hand side of (1) by its mean value form, i. e. , by

$$X_{n+1} = a \cdot (y_n(1 - y_n) + (1 - 2X_n) \cdot (X_n - y_n)) \quad \text{with} \;\; y_n \;\approx\; \text{midpoint of } X_n, \qquad (2)$$

where X_n is an interval in case (ii) and a multiple precision interval in case (iii). Rewriting (1) as (2) does not improve the quality of ordinary floating-point computation, which is still executed using (1).

The following Maple source code uses `intpakX` package to compute orbits for this equation. The results show the approximations x_n obtained by ordinary point evaluations of (1) and the enclosures X_n as obtained by multiple precision interval arithmetic using (1), and (2).

Repeat the computation of x_{1000} using a 10, 15, 20, ... decimal digits arithmetic. The resulting approximations to x_{1000} are all very different. Probably no one is close to the true (mathematical) value. The iteration exhibits chaotic behaviour. What is the true value of x_{1000}?

```
>restart;
>for k from 10 by 5 to 150 do
>   Digits := k;
>   nmax := 1000;
>   a := 3.75;
>   x := .5;
>   for n to nmax do n;
>      x := a*x*(1-x)
>   end do;
>   Digits := 10:
>   print(k, 1.0*x); #leading 10 digits
>end do

number of         approximate
digits used       to x_1000

     10           .7293739044
     15           .8216993992
     20           .4943718351
     25           .8710298086
     30           .8583590713
     35           .6772533909
     40           .8641361707
     45           .8716010232
     50           .9080844468
     55           .6782837431
     60           .7405191406
     65           .8541118421
     70           .5064471708
     75           .7876072081
     80           .8967126257
     85           .8116148942
     90           .4209051185
     95           .8542875712
    100           .8011059764
    105           .8606952069
    110           .9282428337
```

```
115          .7947201576
120          .2817552092
125          .8090124521
130          .3571806913
135          .3516307600
140          .8158439588
145          .8115533519
150          .8464402463
```

Is 8 the first decimal digit of x_{1000}?

Simple idea: all iterates are rational numbers. Why not using Maple's rational number arithmetic?

First we define a Maple function returning the number of decimals used to represent the rational number (number of digits of the numerator and denominator are added):

```
> decDigits:= (x::rational)->
>   ceil(ilog10(numer(x)))
>    + ceil(ilog10(denom(x)))+2:
```

We add two simple procedures to compute the elapsed time:

```
>tic:=proc() global tictoctime:
>   tictoctime:=time(): #start timing
>end proc:

>toc:=proc()    #compute elapsed time
>   time()-tictoctime
>end proc:
```

Now let us try to compute the exact iterates x_n using Maple's rational arithmetic:

```
> tic():
> nmax := 26;
> a := 15/4;  #constant a
> x := 1/2;   #initial value x0
> for n to nmax do
>    x := a*x*(1-x):
>    if n < 4 then
>       print("x:", x);
>    end if;
>    print("n:", n, "digits:",
>    decDigits(x), "time:", toc());
> end do:

nmax := 26
a := 15/4
x := 1/2

"x:", 15/16
"n:", 1, "Digits:",  4, "time:", 0.

"x:", 225/1024
"n:", 2, "Digits:",  7, "time:", 0.

"x:", 2696625/4194304
"n:", 3, "Digits:", 14, "time:", 0.
```

n	digits	elapsed time
4	28	0.
5	57	0.
6	116	0.
7	230	0.
8	462	0.
9	924	0.
10	1849	0.
11	3698	0.
12	7398	0.
13	14796	0.4e-2
14	29592	0.8e-2
15	59184	0.20e-1
16	118370	0.52e-1
17	236739	0.140
18	473479	0.352
19	946958	0.612
20	1893916	1.152
21	3787834	2.152
22	7575668	4.296
23	15151335	10.153
24	30302672	19.193
25	60605342	44.419
26	121210686	86.701

Numerators and denominators become very soon very large integers. To represent x_{26} 121210686 decimal digits are necessary! Also, the computing time increases dramatically. It already takes more than one minute to get x_{26}. The time for computing the next iterate is about twice the time for computing the current iterate. This leads to the conclusion that using Maple it is not possible to compute more than say a few dozen iterates. It is far out of reach to compute (the rational number) x_{1000}. Thus, we still do not know the first correct digits of x_{1000}.

Now let us try to make use of the `intpakX` package. It provides a multiple precision interval arithmetic. Operator symbols denoting the interval operations are $\&+, \&-, \&*$, and $\&/$ (Maple does not allow operator overloading for the basic arithmetical operators).

```
>libname:=
>"/home/kraemer/intpakX/lib", libname
libname := "/home/kraemer/intpakX/lib",
  "/home/kraemer/maple10/lib"
>with(intpakX):       #use the package
>nmax:= 1000;         #enclose x_1000
>Digits:= nmax;       #use 1000 digits
>a:= construct(3.75); #degenerate interval
>x:= construct(0.5);  #degenerate interval
```

```
>tic():               #start timing
>for n from 1 to nmax do
>    x := (a &* x) &* 1 &- x; #interval
>od;                          #expression
>toc();               #elapsed time

>Digits := 10;
>1.0 &* x; #print just a ten-digit enclosure
                nmax := 1000
              Digits := 1000
                   a := [3.75, 3.75]
  x := [0.5, 0.5]
                 1.4746
          [.7917467408, .7917467410]
```

We see a very satisfactory result. Using a 1000 digits multiple precision interval arithmetic `intpakX` shows within one and a half second that $x_{1000} \in [.7917467408, .7917467410]$. Now we see that the first decimal digit of x_{1000} is not 8 but 7. Less mantissa digits would be enough to perform this naive interval computation successfully (try to find the minimal precision!).

As already pointed out: Using a Mean-Value-Form for the iteration function reduces the number of digits needed to get an accurate enclosure pretty much:

```
#Create Mean-Value-Form
> f:= (x,y)-> a*(y*(1-y)+(1-2*x)*(x-y));
#Create corresponding interval expr.
>F:= inapply(f(x), x):
f := proc (x, y)
   options operator, arrow;
   a*(y*(1-y)+(1-2*x)*(x-y))
end proc
>nmax:= 1000;         #comupte x_1000
>Digits:= 180;
>a:= construct(3.75); #degenerat interval
>x:= construct(0.5);
>for n from 1 to nmax do
>   y:= midpoint(x);
>   x:= F(x,y);       #use Mean-Value-Form
>od;
>Digits:= 10;
>1.0 &* x; #ten-digit enclosure
         nmax := 1000
       Digits := 80
            a := [3.75, 3.75]
            x := [.5, .5]
       [.7917467408, .7917467410]
```

Using the Mean-Value-Form a 180 digits interval arithmetic leads to the enclosure $x_{1000} \in [.7917467408, .7917467410]$. Here, some remarks are appropriate:

- Using ordinary multiprecision floating point arithmetic 150 digits are not enough to get the first digit of x_{1000} right (see our fist approach). Using the interval Mean-Value-Form allows the accurate computation of at least the first 15 decimal digits using a 180 digits arithmetic.
- In spite of the fact that compared to the naive interval evaluation we may reduce the precision of the arithmetical operations in the Mean-Value-Form this must not result in faster algorithms. Indeed, our mean-value computation takes about two and a half seconds (compare this with the timing result corresponding to the naive interval evaluation).

Up to now we have seen that multiple precision interval computations allow to verify (long term) orbits of the dynamical system whereas results computed by point computations may be totally incorrect.

Let us now consider the sensivity of orbits with respect to small changes in the inital data. We want to compute two orbits, the first one starting at $x_0 := 0.5$ and the second one starting at $y_0 := x_0 + \varepsilon$. The following interval computations show that using $\varepsilon := 10^{-80}$ the x and y orbits are distinguishable numerically for the iterates at least up to x_{1000} and y_{1000}.

```
> max := 1000;
> Digits := 200;
> a := construct(3.75):
> x := construct(1/2):
> eps := 1e-80;
> y := x &+ construct(eps):
> for n from 1 to nmax do
>    midx := midpoint(x):
>    x := F(x, midx):     #compute x_n
>    midy := midpoint(y):
>    y := F(y, midy):     #compute y_n
> end do:
> Digits := 10:
> 1.0 &* x; #enclosure x_1000
> 1.0 &* y; #enclosure y_n1000

nmax := 1000
Digits := 200
eps := 0.1e-79
[.7917467408, .7917467410]
[.8102867018, .8102867020]
```

It holds $x_{1000} \in [.7917467408, .7917467410]$ and $y_{1000} \in [.8102867018, .8102867020]$. The two intervals are disjoint. Initially nearby points begin to diverge.

Let us now take $\varepsilon := 10^{-90}$. How many leading digits of x_{1000} and y_{1000} are equal? How many digits of the results must must be computed accurately to be able to distinguish the coresponding orbits at the iterates with index 1000? We find the following enclosures: $x_{1000} \in [.79174674092244363768, .79174674092244363770]$ and $y_{1000} \in [.79174674092244363652, .79174674092244363654]$. In both cases the leading

17 digits are guaranteed to be .79174674092244363. Thus, x_{1000} and y_{1000} are distinguishable if at least 18 digits are known accurately. 17 digits are not sufficient for this purpose.

3 Remarks on intpakX

The most important feature of the package is the introduction of new (multiple precision) data types into Maple for *real intervals* and *complex disc intervals*. A range of operators and applications for these data types have been implemented (with separate names), so that the new interval types do not rely on the (rough) standard Maple notion of an interval. Also rounding is done separately to provide the correct rounding mode as needed in interval computation. So, intpakX intervals can be used safely with the implemented operators.

The graphical functions included in intpakX make it more convenient to use Maple graphics for interval computations. They use Maple graphics features to offer special output for the visualization of the intervals resulting from the concerned intpakX functions.

As mentioned above, intpakX defines Maple types for real intervals and complex disc intervals.

On the level of basic operations, intpakX includes the four basic arithmetic operators, including extended interval division as an extra function. Furthermore there are power, square, square root, logarithm and exponential functions (note that square is implemented separately from general multiplication as needed for intervals), a set of standard functions and union and intersection. Reimplementations of the Maple construction, conversion and unapplication functions are added.

As applications, *Verified Computation of Zeros* (Interval Newton Method) with the possibility to find all zeros of a function on a specified interval, and *Range Enclosure* for real-valued functions of one or two variables are implemented, the latter using either interval evaluation or evaluation via the mean value form and adaptive subdivision of intervals. The user can choose between a non-graphical and a graphical version of the above algorithms displaying the resulting intervals of each iteration step.

Additionally, there is a range of operators for complex disc arithmetic. Besides the basic arithmetic operators, there are area-optimal multiplication and division as an alternative to carry out these operations. As a further function, the complex exponential function has been implemented. Range enclosure for complex polynomials serves as an application for complex interval arithmetic.

The package intpakX [4, 7, 8] is freely available on the web. How to install intpakX and how to use this Maple Power Tool in your own worksheets is described in the source files, which are publicly available on the web (see http://www.math.uni-wuppertal.de/~xsc/software/intpakX/). From this link you get the most recent version of inpakX (the version presented by Maplesoft may an older version). In addition, the preprint *Introduction to the Maple Power Tool* intpakX, is available on the web [7]. Its intention is to be a primer to intpakX.

Please note: intpakX assumes that the basic operations provided by Maple are accurate to at least one unit in the last place (ulp) with respect to the actual value of Maple's environment variable Digits (number of decimal digits Mapel uses to represent software floating-point numbers). If so, it is guaranteed that intpakX computes enclosures for ranges of expressions build from these basic operations. Up to now, a violation of the assumption is not known by the authors of intpakX.

A similar statement (1 ulp accuracy) about the accuracy of Maple functions like exp, log, sin, Bessel, etc., is probably not correct. Therefore intpakX uses several guard digits in its computations. Nevertheless computation of expressions involving such functions can not be guaranteed to enclose the true range! Despite this limitation, we are convinced that intpakX is a valuable didactical tool to illustrate interval algorithms/methods. As soon as error bounds for Maple funcions will be available, intpakX will use the bounds to compute guaranteed enclosures.

4 Conclusion

It may be a hard job to compute accurate numerical results in the field of chaotic systems. Traditional (multiple precision) floating point arithmetics do not give guaranteed results. Even with a lot of experimental work using different precisions it is not possible to get numerical results with mathematical rigor. In contrast with this unsatisfactory situation, using a multiple precision interval arithmetic leads to guaranteed results. Our Maple power tool intpakX offers among others such an arithmetic. In this paper we have used this package successfully to get some rigorous mathematical statements about the chaotic behaviour of the logistic map.

References:

[1] R. Devaney, An Introduction to Chaotic Dynamical Systems, Menlo Park, CA: Ben-

jamin/Cummings, 1986.

[2] S. Elaydi, Saber, Discrete Chaos, Boca Raton: Chapman & Hall, 2000.

[3] P. Giesl, Construction of Global Lyapunov Functions Using Radial Basis Function, LNM 1904, Springer, 2007.

[4] M. Grimmer, Interval Arithmetic in Maple with intpakX, PAMM - Proceedings in Applied Mathematics and Mechanics, vol. 2, 2003, pp. 442–443.

[5] W. Krämer, U. Kulisch, and R. Lohner, Numerical Toolbox for Verified Computing II - Advanced Numerical Problems. Draft version, available from http://www.uni-karlsruhe.de/~Rudolf.Lohner/papers/tb2.ps.gz

[6] Introduction to the Maple Power Tool intpakX, Preprint BUW WRSWT 2006/9, University of Wuppertal, 2006, pp. 1–31. To appear in: Bulgarian Serdica Journal of Computing.

[7] W. Krämer, Introduction to the Maple Power Tool intpakX, Preprint BUW WRSWT 2006/9, University of Wuppertal, 2006, pp. 1–31. To appear in: Bulgarian Serdica Journal of Computing.

[8] W. Krämer, intpakX - An Interval Arithmetic Package for Maple, accepted for publication, proceedings to SCAN 2006, IEEE CPS (Conference Publishing Service), 2007.

[9] R.M. May, Simple mathematical models with very complicated dynamics, Nature 261: 459, 1976.

[10] T. Nicholas, T. Abbott, J. Reilly, An experimental approach to nonlinear dynamics and chaos, Addison-Wesley New York, 1992.

[11] G. Osipenko, Dynamical Systems, Graphs, and Algorithms. LNM 1889, Springer, 2007.

[12] S. Strogatz, Nonlinear Dynamics and Chaos, Cambridge MA: Perseus, 1994.

[13] The free encyclopedia, Mai 2007. http://en.wikipedia.org/wiki/Logistic_equation

[14] http://www.math.uni-wuppertal.de/~xsc/software/intpakX/