Finding and proving the existence of periodic orbits and their Conley-Zehnder index on a computer

Otto van Koert

Department of Mathematical Sciences, Seoul National University

Goals

The main goal is to give an introduction to the basics of rigorous numerics. Specifically, we

- describe some methods to prove existence of periodic orbits using a computer. These methods construct a neighborhood of the orbit.
- give a method to determine the Conley-Zehnder index on the computer

Remark

The methods involved are old and standard for those who know them. Some names involved with the development of the methods are: Moore, Lohner, Simó, Tucker, Zgliczyński, many others

The standard library is CAPD (computer assisted proofs in dynamics), although I used my own here.

One basic idea to find periodic orbits

Traditionally, the following is tried and tested method

1. find a local surface of section transverse to the flow (this can be hard)
2. obtain the locally defined return map f by numerical integration
3. use Newton iterations to find zeroes of $f(x) - x$; if x were really a zero of the true return map, then this x is the starting point of a periodic orbit.

We run into obvious problems when we want to be sure:

- Step 2 only gives an approximation; how large is the error?
- Even if we really have the exact f, step 3 only gives approximations of a zero;
- Floating point arithmetic is non-exact by nature, and provides an additional source of errors.

Rounding errors: how bad can they get?

We start with a (famous?) example taken from the book of Moore: "Interval analysis". Consider the rational function

$$f(a,b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

We want to know $f(77617.0, 33096.0)$.
According to Moore et al, an IBM 370 returns the following

single precision : 1.17260361...

double precision : 1.17260394005317847...

extended precision : 1.17260394005317863185...

However, the actual answer is $-0.827396...$

Remark

Only the double precision reproduces on a modern computer: still, modern computers don't do better.

A recent, non-contrived example

Remark

Computing with enough decimals always solves the problem, but we don't know in advance how many we need (unless we are extremely careful with every single formula).

Verifying in maple whether a determinant is non-zero. (for details see Maple session)
With default settings maple returns the answer

$$-1.633034062 \cdot 10^{12} < 0.$$

However, the true answer is positive.

Interval arithmetic

Compute with (ideally small) sets rather then just with points.

- Define operations $\oplus$, $\otimes$, $\tilde{\exp}$, etc, that "enclose" arithmetic operations $+$, $\cdot$, exp. This means

$$\begin{aligned} [a,b]+[c,d] &\subset [a,b]\oplus[c,d], \\ [a,b]\cdot[c,d] &\subset [a,b]\otimes[c,d], \\ [\exp a, \exp b] = \exp([a,b]) &\subset \tilde{\exp}([a,b]). \end{aligned}$$

- On a computer one needs to take care of rounding errors: intuitively, this is done by systematically rounding down and rounding up.

Example

For example $[0,\pi]\subset[0,3.1416]$. So with 5 digits, we should get

$$\widetilde{\sin([0,\pi])} = [-0.73465\cdot 10^{-5}, 1],$$

or an even larger interval.

Good news:

- Interval arithmetic combined with arbitrary precision computations (more than extended precision) will solve the above problems
- there are several good libraries available.

Bad news:

- there is a trend in modern processors to make numerical computations faster by allowing small errors. These hardware speedups should be avoided in reliable interval libraries.
- even the compiler matters.
- upshot: multiprecision interval arithmetic is very slow.

Dependency problem

Operations with interval arithmetic often leads to overestimation of the enclosing sets:

- With interval arithmetic, we find

$$[-1,1]-[-1,1]=[-2,2],$$

even though for all $x \in [-1,1]$ we have $x-x=0$.
- if $f(x)=x^2-x$, then $f([-1,1]) \subset [-1/4,2]$, but with direct interval arithmetic we get

$$\tilde{f}([-1,1])=[-2,2],$$

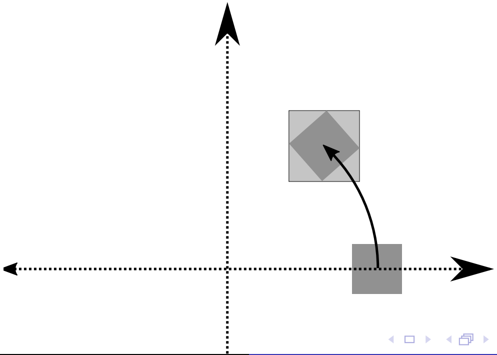
which overestimates the interval.

Wrapping effect

Consider the ODE

$$\dot{q} = p$$
$$\dot{p} = -q$$

and start with a box $[1-\delta, 1+\delta] \times [-\delta, \delta]$. After time t (assume $t \notin \frac{\pi}{2}\mathbb{Z}$), the box has rotated, and a new enclosing box has grown by a factor, even with exact arithmetic.



This effect is known as the wrapping effect, and causes obvious problems: even with such a simple, well-behaved equation, the box grows exponentially in size:

1. Naive interval arithmetic will only work for relatively short orbits
2. One can try to combat this effect by different representations of the set: ellipsoids, and rotated rectangles will alleviate the problem to some degree.
   These methods, in particular in combination with ODEs, were pioneered by Lohner in the late 80's
3. Also, one can subdivide a big box into many smaller boxes to reduce.

Remark

We will mostly ignore this (important) issue in this talk.

Numerical integration of ODEs

There are many schemes to perform numerical integration.

- Runge-Kutta methods: traditionally the error can be estimated by extrapolation.
- geometric integrators. For example symplectic integrators. These preserve the symplectic form when using exact arithmetic.
- Taylor method.

Remark

- *In principle, any of these methods can be used provided we have an explicit formula for an error bound.*
- *In practice, we want a large stepsize (to make fewer rounding errors) combined with high precision.*

Taylor method

We want to approximate solutions to the initial value problem

$$\dot{x} = f(x)$$
$$x(0) = x_0$$

with explicit error control. We assume that f is analytic.
Differentiating the equation, we find

$$x^{(k)} = \nabla_f \dots \nabla_f f.$$

Observation: $x^{(k)}(t)$ is determined by $x(t)$
With Lagrange's formula for the remainder of a Taylor approximation we find the errors, but this requires an unknown $\tau \in]0, t[$.
We will enclose the error $R_k(t)$ using interval arithmetic, the above observation and Lagrange's formula.

To control the error, we roughly fix the stepsize and vary the order. We choose the order so large that the the remainder is smaller than the desired error (for example "smaller than the rounding error").

Remark

This does not imply that the error is smaller than the rounding error. Instead, it only means that the rounding errors are the main source of errors.

Automatic differentiation versus symbolic differentiation

The most obvious way to obtain all higher derivatives

$$x^{(k)} = \nabla_f \dots \nabla_f f$$

is to perform symbolic differentation, either by hand or with the computer, yielding a function in the initial data. This works extremely well for the harmonic oscillator.
However, the number of terms in higher derivatives increases extremely quickly for (most) ODE's.

Example

For the planar three-body problem with three equal masses, the number of terms (monomials) roughly multiplies with a factor 8 each time the order is increased by 1; it requires 220MB to just store the functions in the initial data that describe the derivatives up to order 7. This phenomenon is known as **expression swell**.

The way out is to drop the demand for an explicit formula. To do so, consider a standard way in which a computer can manipulate a formula: the expression tree.

- construct a tree in which each vertex is an operator
- nullary operators: constants, variables
- unary operator: functions
- binary operators: $+, \cdot$, etc

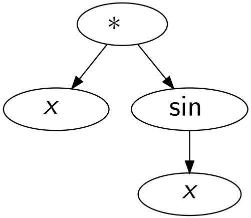
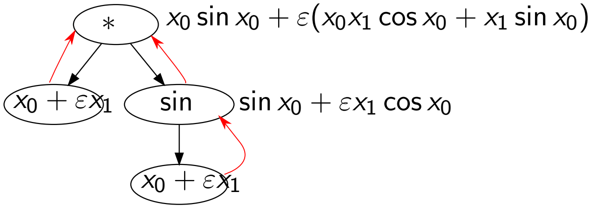


Figure: An expression tree for $x \sin(x)$

Simply put, the idea for automatic differentation is to propagate the derivatives from the bottom (forward mode), or from the top (backward mode: good for gradients), and get numbers rather than formulas: this keeps the complexity under control.
An easy way to implement this uses **dual numbers**: We work in the ring $\mathbb{R}[\varepsilon]/\langle\varepsilon^2\rangle$ to encode both the value and the derivative of a function.



$*$
$x_0 \sin x_0 + \varepsilon(x_0 x_1 \cos x_0 + x_1 \sin x_0)$
$x_0 + \varepsilon x_1$
sin
$\sin x_0 + \varepsilon x_1 \cos x_0$
$x_0 + \varepsilon x_1$

Figure: Propagating the derivatives from the bottom. By immediately inserting values, things are kept simple.

For higher order derivatives, we use jets. Up to order k, this means that we work in the ring $\mathbb{R}[\varepsilon]/\langle\varepsilon^{k+1}\rangle$. This can be used to encode the degree k Taylor polynomial of a function.

1. In practice we only need to teach the computer how to add, subtract, multiply and divide in the ring $\mathbb{R}[\varepsilon]/\langle\varepsilon^{k+1}\rangle$.
2. If we have transcendental functions or roots, then these also need to be included.
3. Once this is done, the Taylor polynomial of any composition of these operations is automatically determined.

Simple (and slow) implementation of the Taylor method for ODE's

Input: ODE of the form $\dot{x} = f(x)$, $x(0) = x_0$.

Output: Taylor polynomial of $x(t)$ up to order k

1. Write $x(t) = x_0 + x_1 t + x_2 t^2 + \ldots$. Note
$$\dot{x}(t) = x_1 + 2x_2 t + 3x_3 t^2 + \ldots = f(x(t))$$
2. Expand $f(x(t))$ "recursively" to get $f_0 + f_1 t + f_2 t^2 + \ldots$.
3. To do so, compare left and right hand side: f_i (determined by $x_0, \ldots x_{i-1}$) determines x_i via the ODE

One can use repeated insertion to obtain the jet up to order k.

Rough enclosure

We consider the initial value problem

$$\begin{aligned} \dot{x} &= f(x) \\ x(0) &= x_0 \end{aligned}$$

Lemma (Moore-Lohner rough enclosure)

Suppose $[u_0]$ and $[u]$ are boxes such that

$$[u_1] := [u] + [0,h]f([u_0]) \subset [u_0]$$

Then for every initial value $x_0 \in [u]$, the solution $x(t)$ to the above IVP exists on $[0,h]$ and is contained in $[u_1]$.

This allows us to evaluate the bound the remainder term by enclosing the $n+1$-derivative on the set $[u]$ (or $[u_0]$).
This can be done by applying interval arithmetic.

Simple Taylor step algorithm with enclosure

Input: an enclosure $[u]$ of the set of initial values. A (representable) stepsize h, and a truncation precision ε, a maximum order N.

Output: Returns failure if the truncation error cannot be made smaller than ε with order less than N. Otherwise returns an enclosure of $Fl_h^X([u])$, the time h-flow of the vectorfield X, such that the truncation error is smaller than ε.

1. Find a rough Lohner-style enclosure $[u_0]$
2. Enclose $(\nabla_f \dots \nabla_f f)(x)$ for $x \in [u_0]$. Obtain an error vector

   $$[z] = [\frac{[0,h]^{n+1}}{(n+1)!}[\nabla_f \dots \nabla_f f)([u_0])]]$$

   Choose n so large that $[\|z\|] < \varepsilon$. Return failure if this is not possible with $n < N$
3. Return an enclosure of $Fl_h^X([u])$ using the Taylor polynomial

   $$[P_n[u](h) + z],$$

More precisely, $P_n[u]$ is an enclosure of the Taylor polynomial at h for initial values in $[u]$.

Remark

To reduce the wrapping effect, steps 2 and 3 need to done more carefully (for example with coordinate transformations).

Remark

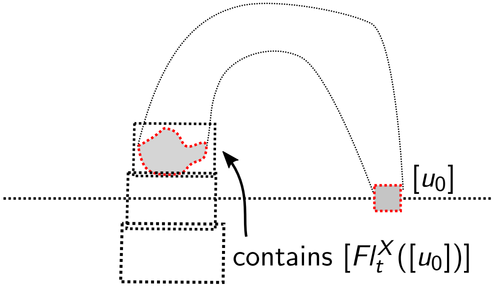
The jet/Taylor polynomial approximation are all computed with interval arithmetic.

Validation of periodic orbits (simple approach)

Suppose (x_0, t_0) is a candidate for a periodic orbit on a local surface of section L, typically obtained by non-rigorous techniques (no interval arithmetic).

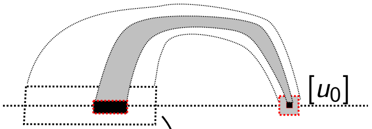
Goal: Find a neighborhood $([x_0], [t_0])$ such that $[x_0]$ contains the starting point of a periodic orbit with period $T \in [t_0]$.

We shoot until we hit the surface of section: we need boxes before, after and during crossing.



For symmetric orbits, we can surround the slope with boxes to left and to the right (intermediate value argument: but regard it as a degree argument).
More systematically, use the enclosed Taylor method for the variational equations.

- we get enclosures $[slope([t_{cross}])]$, and $[slope'([t_{cross}])]$
- if $[slope'([t_{cross}])]$ is almost constant (which is true for small boxes by continuity), we can conclude existence of a zero by shooting again with slightly larger box.



Degree arguments work in general: in practice Interval Newton

Theorem (Alefeld)

Suppose $f : \mathbb{R}^n \to \mathbb{R}^n$ is a smooth function. Let X be a box. Assume $[df(X)]$ is invertible. For $x_0 \in X$ put

$$N(x_0, X) = x_0 - [df(X)]^{-1} f(x_0)$$

Then

1. *f is injective on X*
2. *if $N(x_0, X) \subset X$, then there is a unique $x^* \in X$ such that $f(x^*) = 0$*
3. *if $x_1 \in X$ and $f(x_1) = 0$, then $x^* \in X$*
4. *if $N(x_0, X) \cap X = \emptyset$, then $0 \notin f(X)$*

This interval Newton operator can be enclosed with the enclosed Taylor method: apply it to the return map minus identity.

Conley-Zehnder index

Recall that the Conley-Zehnder index of a path of symplectic matrices $\Psi : [0, T] \to Sp(2n)$ is the "algebraic intersection number" with the Maslov cycle

$$V = \{A \in Sp(2n) \mid \det(A - \mathrm{Id}) = 0\}$$

For paths in $Sp(2)$ we can compute this index using the rotation number.
For a periodic orbit, this path Ψ is obtained by measuring the linearized flow with respect to a suitable frame (for example a global frame constructed with quaternions).

To obtain the rotation number, we follow the path of symplectic matrices by retracting it to $U(1)$.

- compute the linearized flow using the variational equations (also with the above method)
- the rough enclosures and boxes around the sequence $\Psi(t_i)$ ensure we get an enclosure of the rotation number.
- if $[\Psi(t_{final})]$ contains no eigenvalue equal to 1 (dimension 2: check trace), then the Conley-Zehnder index is defined, and can be computed with Long's formula.

To do this, we apply the enclosed Taylor method to the (first) variational equation.

Sample theorems

So far, only very modest results:

Theorem (Retrograde orbit in RTBP)

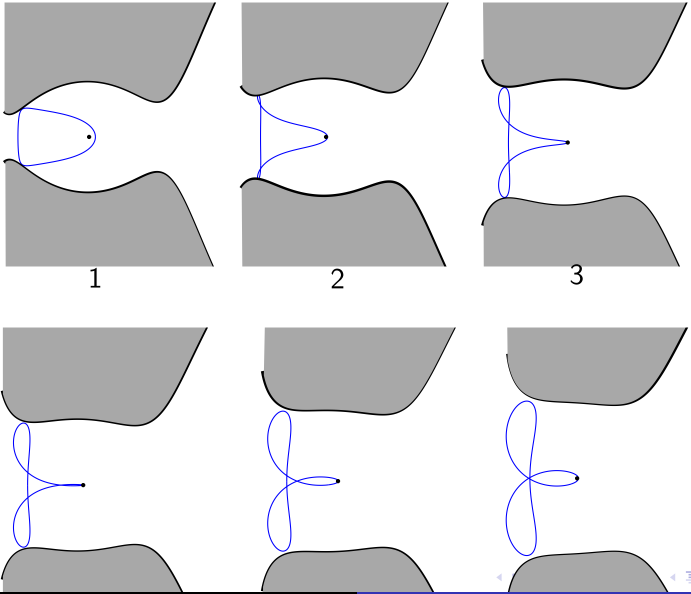
The Birkhoff orbit is non-degenerate and has Conley-Zehnder index equal to 1 *in the Moser regularization for all* $c \in [2.0, 2.001]$ *and* $\mu \in [0.125, 0.1255]$.

Just run the code for this parameter range. The enclosure of the trace of the linearized return map lies does not contain 2, so these orbits are all non-degenerate and have equal index

Remark

This takes about 40s with unoptimized code (non-rigorous, slightly optimized code takes only 0.01s). Repeating this for other parameter values only takes computation time, but I haven't done this. Lower precision will also run much faster and can produce the same result.

A family of symmetric orbits in RTBP ($\mu = 0.99$) all with $\mu_{CZ} = 3$ whose double covers are bad.



Theorem

There is a symmetric direct orbit for $\mu = 0.99$ and $c = 1.57$ with initial value q_1 in small interval (width 10^{-14}). This orbit is a negative hyperbolic orbit (so its double cover is bad) with Conley-Zehnder index 3. This orbit is part of a 1-parameter family in c which is elliptic for $c > 1.58$.

Upshot:

- Periodic orbits can be found and their CZ-indices are computable
- (not explained here) given a action bound L, we can in principle find all periodic Reeb orbits on a starshaped hypersurface with action less than L
- Floer differential: I have no idea.