Automatic Differentiation methods in computational Dynamical Systems: invariant manifolds and normal forms

Àlex Haro

Departament de Matemàtica Aplicada i Anàlisi
Universitat de Barcelona

IMA (University of Minnesota), June 2011

Preface
Poincaré's program

Poincaré's program for global analysis of dynamical systems (DS):

- Identify equilibriums and periodic orbits.
- Identify the invariant manifolds associated to them.
- Identify other invariant manifolds (e.g. invariant tori).

The long term behaviour of a dynamical system is organized by its invariant objects.

These objects constitute the skeleton of the DS.

Nowadays: The qualitative approach can be made very quantitative!

Preface

My (and your!) goals of the lectures

- Describe a unified framework for the computation of invariant manifolds and normal forms attached to fixed points of vector fields using power series.
- Adapt the methodology to *elementary* models. By *elementary* we mean that the model can be written using arithmetic operations and elementary functions.
  - Introduce methods of Automatic Differentiation.
  - Estimate complexities.
- Apply the methodology to a concrete example: center manifolds of Lagrange equilibrium points in the Restricted Three Body Problem.

Preface
Sources of inspiration

Poincaré, Arnold, ...

- C. Simó, “On the Analytical and Numerical Approximation of Invariant Manifolds” (1990)
- X. Cabré, E. Fontich, R. de la Llave, “The parameterization method for invariant manifolds” (2003)
- À. Jorba,“A methodology for the numerical computation of normal forms, centre manifolds and first integrals of Hamiltonian systems” (1999)
- D. E. Knuth, “The art of computer programming. Vol. 2: Seminumerical algorithms” (3rd rev., 1997)

Notes of the course:

`http://www.ima.umn.edu/2010-2011/ND6.20-7.1.11/abstracts.html#11183`

Table of contents

Functional equations in Dynamical Systems

Functional Equations in Dynamical Systems

Some examples and methodologies

Examples:

- The solution of odes
- The normal form equations
- The invariance equations for invariant manifolds

Methodologies:

- Expand the unknowns using series (Taylor, Fourier, Fourier-Taylor), approximate them using interpolants (polynomials, splines, etc.)
- Solve the equations “order by order” (on-line methods for power series)
- Solve the equations using e.g. Newton method

Remark:

- We have to evaluate the dynamical system on the series!

These tasks depend a lot on the problem.

Invariant manifolds for maps

Setting the equations

Given:

- a map in $\mathbb{R}^n$:

$$\bar{z} = F(z)$$

- a d-manifold $\mathcal{W}$, parameterized by

$$z = W(s),$$

  where $W : U \subset \mathbb{R}^d \to \mathcal{W} \subset \mathbb{R}^n$

- a map in $\mathcal{W}$, written in coordinates on $U \subset \mathbb{R}^d$ as

$$\bar{s} = f(s)$$

then:

The manifold $\mathcal{W}$ parameterized by W is invariant under F, with subsystem given by f, if

$$F(W(s)) = W(f(s)) \ .$$

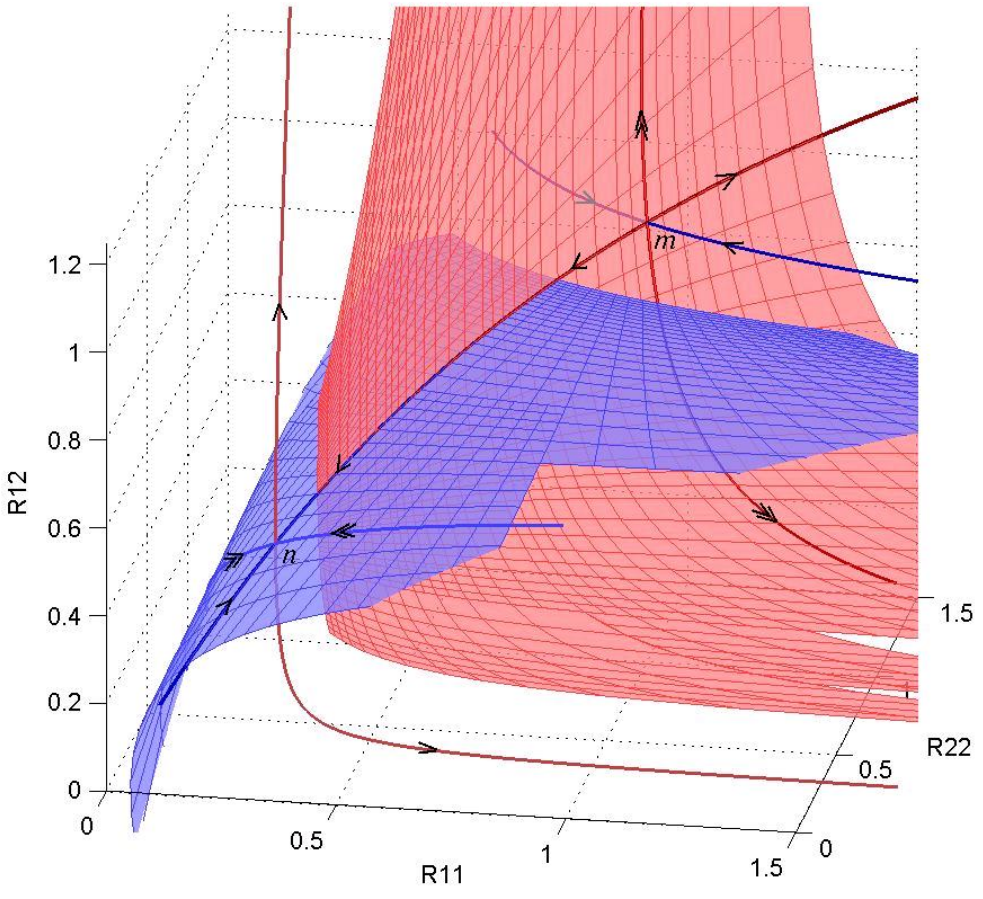
Invariant manifolds for maps

Setting the equations

Pictures at an exhibition

Stable and unstable manifolds in a 4D economic growth model

$$
\begin{aligned}
R_{(1,1),t+1} &= \frac{1}{1+\alpha-\alpha R_{(2,1),t}} \\
R_{(2,2),t+1} &= \frac{1}{1+\beta-\beta R_{(1,2),t}} \\
R_{(1,2),t+1} &= R_{(2,2),t+1} \cdot \frac{R_{(2,2),t}}{R_{(2,1),t}} \\
R_{(2,1),t+1} &= R_{(1,1),t+1} \cdot \frac{R_{(1,1),t}}{R_{(1,2),t}}
\end{aligned}
$$

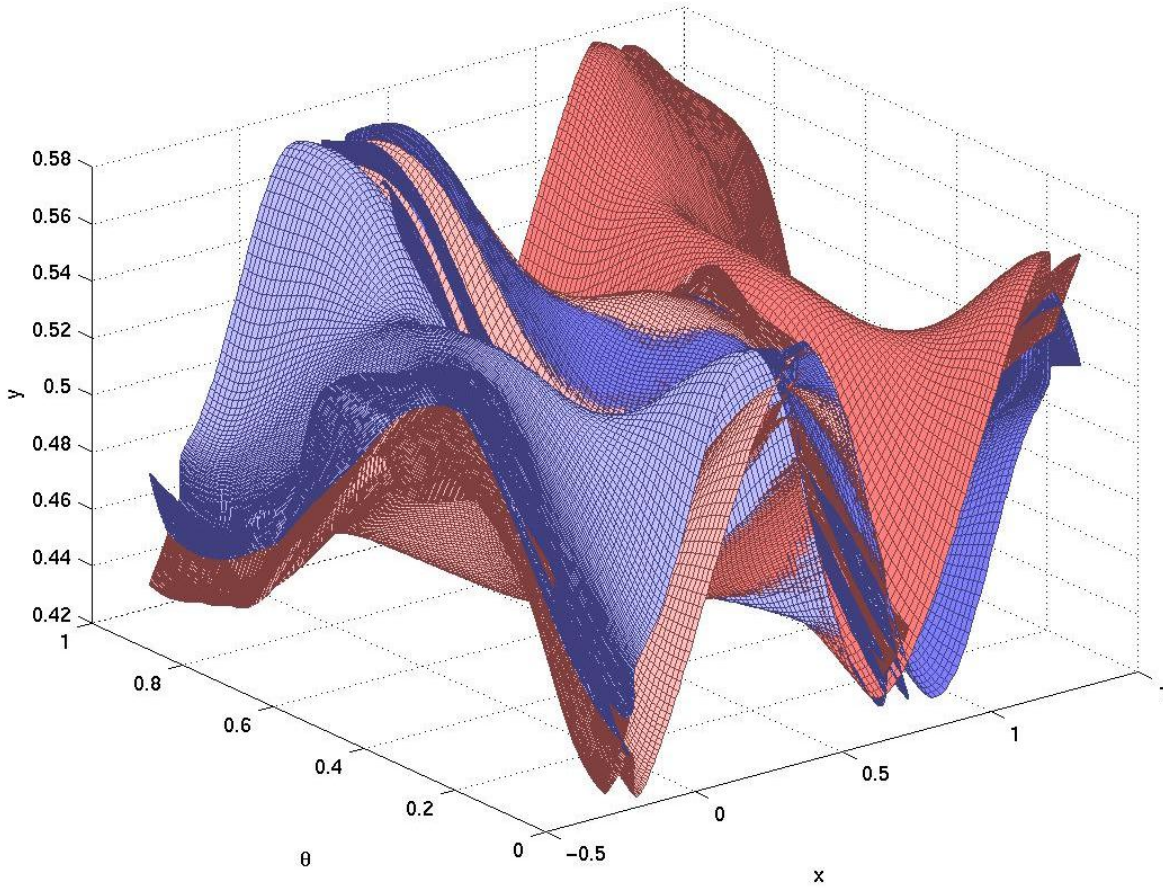


(with Pere Gomis-Porqueras)

Pictures at an exhibition

Stable and unstable manifolds of a 2-period torus in a qp driven system

$$
\begin{aligned}
\bar{x} &= x + \bar{y} \mod 1 \\
\bar{y} &= y - \frac{\sin(2\pi x)}{2\pi}(\kappa + \lambda \cos(2\pi\theta)) \\
\bar{\theta} &= \theta + \omega \mod 1
\end{aligned}
$$

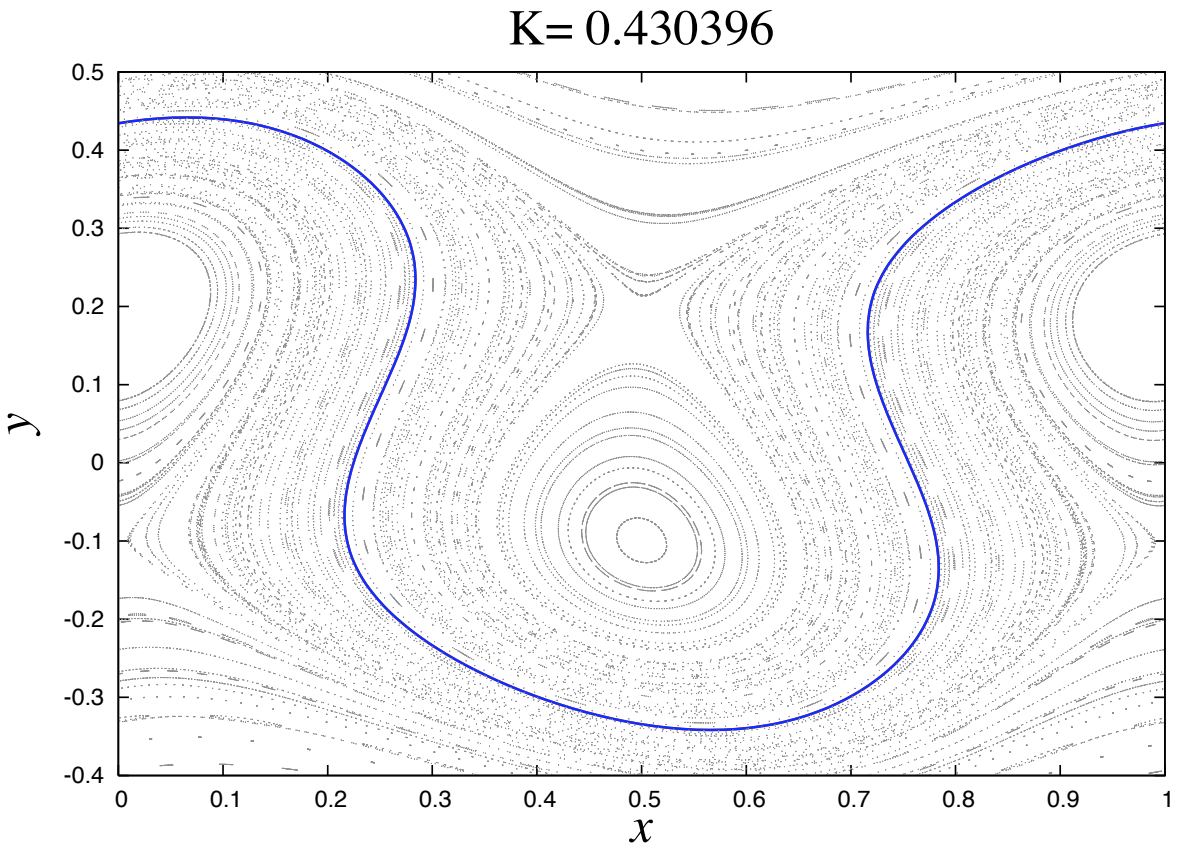


with R. de la Llave

Pictures at an exhibition

Meandering KAM tori in a degenerate area preserving map

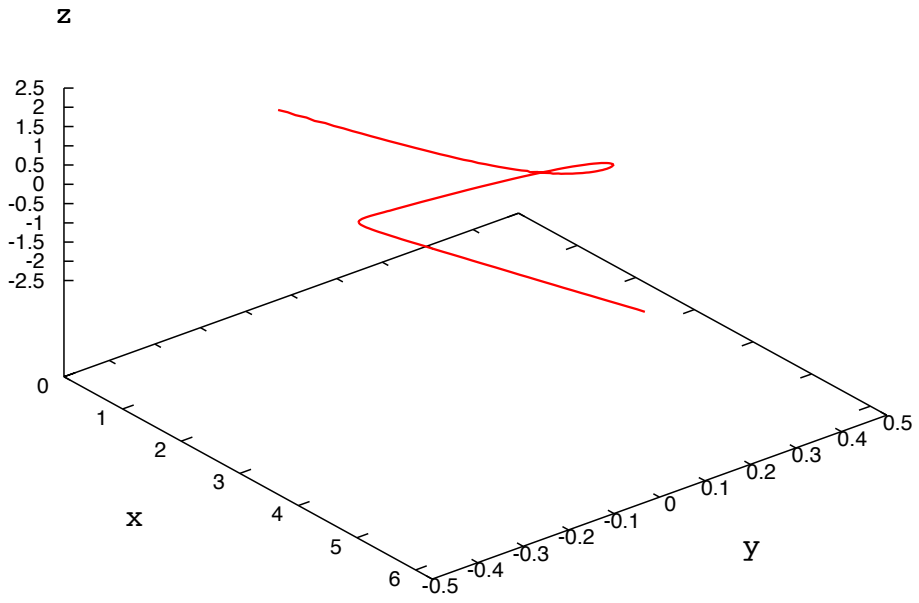
$$\begin{cases} \bar{x} &= x + (\bar{y} + 0.1)(\bar{y} - 0.2)\,, \\ \bar{y} &= y - \dfrac{\kappa}{2\pi} \sin(2\pi x)\,. \end{cases}$$

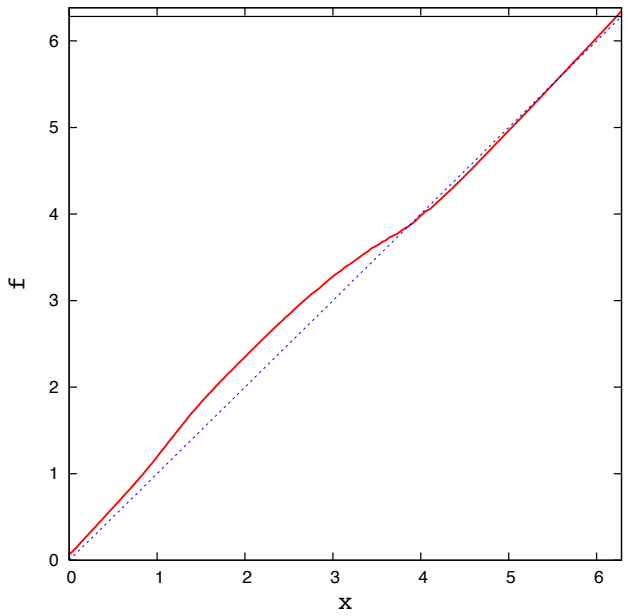


with R. de la Llave and A. González

Pictures at an exhibition

Normally hyperbolic tori in a non-conservative system





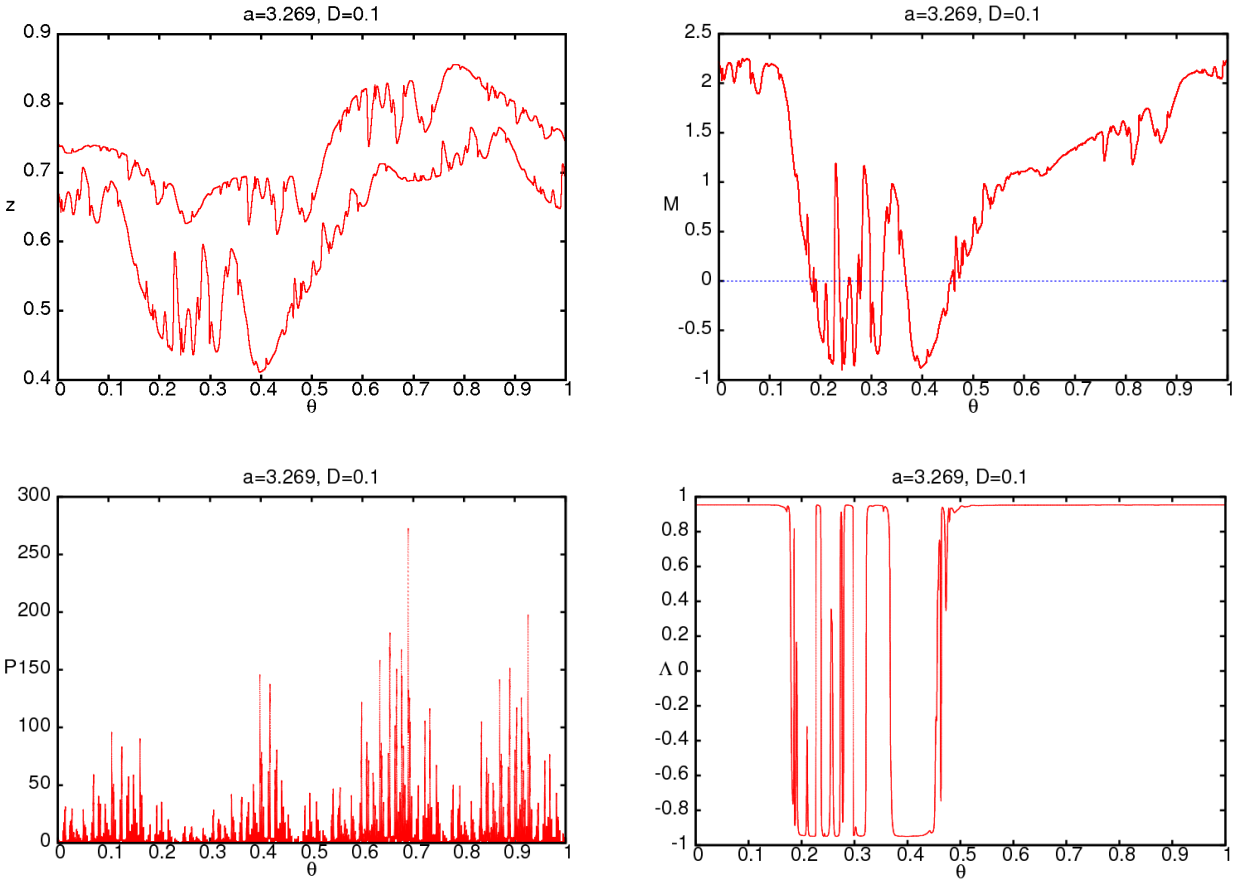
Invariant torus and dynamics in the fattened Arnold family:

$$\bar{x} = x + a + \varepsilon(y + \frac{z}{2} + \sin x)$$
$$\bar{y} = b(y + \sin x)$$
$$\bar{z} = c(y + z + \sin x)$$

with $a = 0.1$, $b = 0.3$, $c = 2.4$, $\varepsilon \simeq 0.750396$. (with Marta Canadell)

Pictures at an exhibition

Computer assisted proofs on the verge of a hyperbolicity breakdown



Invariant tori and normal dynamics of a 2 period torus of the qp driven logistic map

$$\begin{aligned} \bar{z} &= a(1 + D\cos(2\pi\theta))z(1 - z) \\ \bar{\theta} &= \theta + \omega, \end{aligned}$$

where $\omega = \frac{\sqrt{5}-1}{2}$, and a and D are parameters. (with Jordi-Lluís Figueras)

Invariant manifolds for vector fields

Setting the equations

Given:

- a vector field in $\mathbb{R}^n$:

$$\dot{z} = F(z)$$

- a d-manifold $\mathcal{W}$, parameterized by

$$z = W(s),$$

  where $W : U \subset \mathbb{R}^d \to \mathcal{W} \subset \mathbb{R}^n$

- a vector field in $\mathcal{W}$, written in coordinates on $U \subset \mathbb{R}^d$ as

$$\dot{s} = f(s)$$

then:

The manifold $\mathcal{W}$ parameterized by W is invariant under the flow of F, with subsystem given by f, if

$$F(W(s)) = \mathrm{D}W(s)\, f(s)\ .$$

Functional equations

Unknowns

In the invariance equation:

- There is a known term, F.
- There are two unknowns, W and f.

The system is underdetermined:

- There are n equations.
- There are $n + d$ unknown d-variate functions.

Functional equations

Particular setting

Setting the problem

Given a fixed point z^0, with a d-dimensional invariant subspace for the linearization $\dot{z} = \mathrm{D}F(z^0)z$ around z^0, associate an invariant manifold tangent to such subspace.

Semi-local analysis. We will expand the manifold (and its dynamics) using power (or Taylor) series.

Globalization. The semi-local analysis is the first step to globalize the manifold, that is to extend the manifold far away of the fixed point.

- 1D stable/unstable manifolds is straigtforward.
- 2D stable/unstable manifolds, see [KrauskopfODHGVDJ05].
- High order approximations are crucial for center manifolds.

Functional equations

Composition with the system

Important fact

The term $F(W(s))$ involves compositions of the system with the parameterization.

Many times the model F is *elementary*, and the compositions can be made very efficiently!

Automatic Differentiation and Multivariate power series

Algebraic Manipulation

Series as algebraic objects

Observation: series are algebraic objects.

Algebraic Manipulation

AM is a set of techniques based on the mechanical application of algebraic operations on series, such as arithmetic operations and compositions.

Some AM `C/C++` sofware: `TRIP`, `Piranha`.

This approach has a long story in Celestial Mechanics!

Automatic Differentiation (AD)

Coefficients and derivatives

Observation: coefficients of power series expansions are (normalized) derivatives.

Automatic Differentiation (AD)

AD is a set of techniques based on the mechanical application of the chain rule to obtain derivatives of a function given as a computer program.

AD is a technology for automatically augmenting computer programs, including arbitrarily complex simulations, with statements for the computation of derivatives, also known as sensitivities.
`http://www.autodiff.org`

Some AD `C/C++` sofware: `ADOL-C`, `COSY INFINITY`, `C-XSC`.

Automatic Differentiation and Algebraic Manipulation

Merging the two technologies

Derivatives of arithmetic operations of power series can be performed easily from the corresponding algebraic operations.

Derivatives of composition of power series with elementary functions can be performed on-line (order by order), à la Knuth.

Multivariate power series

Definitions and notations

Definition

Given a commutative ring $\mathbb{K}$ (like $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$), a (formal) power series in the variables $x = (x_1, \ldots, x_d)$ with coefficients in $\mathbb{K}$ is an element of the ring $\mathbb{K}[[x]] = \mathbb{K}[[x_1, \ldots, x_d]]$,

$$f(x) = \sum_{k=0}^{\infty} f_k(x) \,, \tag{1}$$

where each $f_k(x)$ is a homogeneous polynomial of order k, that is an element of $\mathbb{K}_k[x]$, which we write

$$f_k(x) = \sum_{m_1+\cdots+m_d=k} f_{m_1,\ldots,m_d} x_1^{m_1} \ldots x_d^{m_d} = \sum_{|\mathbf{m}|=k} f_{\mathbf{m}} x^{\mathbf{m}}$$

We also denote the kth truncation as $f_{\leq k}(x) = \sum_{i=0}^{k} f_i(x)$.

Multivariate power series

Homogeneous polynomials

Recursive scheme

A homogeneous polynomial $f_k(x)$ of d variables $x = (x_1, \ldots, x_d)$ of order k is a combination of $(k+1)$ homogeneous polynomials of the first $(d-1)$ variables $\hat{x} = (x_1, \ldots, x_{d-1})$ of degrees $k, k-1, \ldots, 0$:

$$f_k(x) = f_k^d(\hat{x}) + f_{k-1}^d(\hat{x})x_d + \cdots + f_0^d(\hat{x})x_d^k \ .$$

A kth order homogeneous polynomial is represented as:

- A vector of $\mathrm{h}_d(k) := \binom{d+k-1}{d-1}$ coefficients, in *graded reverse lexicographic ordering* w.r.t. the keys $\mathbf{m} = (m_1, m_2, \ldots, m_d)$.
- A recursive tree.

A kth truncated power series has $\mathrm{n}_d(k) := \binom{d+k}{d}$ coefficients.

Some implementation details

Storage of coefficients

Two main approaches

- For *dense* power series, one stores *all* the coefficients.
- For *sparse* power series, one stores the non-zero terms, corresponding to coefficient-key pairs.

Here: dense polynomials, without structure.

TRIP and Piranha supports sparse series.

Some implementation details

Graded reverse lexicographical ordering

Observation: The ordering of the coefficients is induced by the recursive tree scheme.

Example

The 10 coefficients of a homogenous 3-variate polynomial of order 3 are ordered following the scheme:

$$x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3, x_1^2 x_3, x_1 x_2 x_3, x_2^2 x_3, x_1 x_3^2, x_2 x_3^2, x_3^3.$$

Some implementation details

Data structures

```
1  // structure for homogeneous terms
2  // the coefficients are double type
3
4  struct homog
5  {
6    unsigned char nv, orden; // number of variables, homogeneous order
7    double *coef; // address of the first coefficient
8    struct homog *term; // for tree distribution of coefficients
9  };
10
11 typedef struct homog homog;
12
13 // structure for power series
14
15 typedef struct
16 {
17   unsigned char nv, orden; // number of variables, order of the series
18   homog **term; // list of homogeneous terms
19 }
20 serie;
```

Product of (truncated) power series

Schoolbook formula

The key routine, on which the rest of routines are built, is the (truncated) *product* of power series, that reduces to perform products of homogeneous polynomials.

We use the naive convolution formula, whose cost (number of operations) is

$$\mathrm{p}_d(k) = \binom{2d+k}{2d} \sim \frac{1}{(2d)!}k^{2d} \sim \frac{d!^2}{(2d)!}\mathrm{n}_d(k)^2\,.$$

By *operation* we mean "one multiplication of coefficients, one addition, and index the result".

Important:

- The method is on-line.
- Easy to implement.

Product of (truncated) power series

Fast methods

There are (asymptotically) fast convolution algorithms

- Karatsuba and Toom-Cook methods, with cost $\sim n^{\alpha}$, with $\alpha < 2$, $n = \mathrm{n}_d(k)$.
- FFT methods, with multi-evaluation and interpolation techniques and reduction to univariate series, with cost $\sim n \log(n)$, with $n = \mathrm{n}_d(k)$.

Important:

- useful only for very high order (not usual in DS).
- multi-evaluation and interpolation produces numerical errors.

Some implementation details

Scalar multiplication of a homogenous polynomial

Natural code

```
1  void smulth (homog *h, double r, homog *a)
2  {
3    unsigned int m, nch;
4
5    nch= nch(h->nv,h->orden);
6
7    for (m= 0; m<nch; m++){
8      h->coef[m]+= r* a->coef[m];
9    }
10 }
```

Fancy code

```
1  void smulth (homog *h, double r, homog *a)
2  {
3    double *hc, *hf, *ac;
4
5    hf= h->coef + nch(h->nv,h->orden);
6    for (hc= h->coef, ac= a->coef; hc<hf; (*hc)+= r* (*ac), hc++, ac++);
7  }
```

Some implementation details

Product of homogeneous polynomials

```
1  void sprodh (homog *p, homog *a, homog *b)
2  {
3    if (p->nv>2) {
4      if (a->orden) {
5        if (b->orden) {
6          homog *aa, *bb, *pp, *pp0, *af, *bf;
7          af= a->term+a->orden; bf= b->term+b->orden;
8          for (aa= a->term, pp0= p->term; aa<af; aa++, pp0++) {
9            for (bb= b->term,pp= pp0; bb<bf; sprodh(pp, aa, bb), bb++, pp++);
10           smulth (pp, *(bb->coef), aa);
11         }
12         for (bb= b->term, pp= pp0; bb< bf; smulth(pp, *(aa->coef), bb), bb++, pp++);
13         *(pp->coef)+= *(aa->coef) * *(bb->coef);
14       }
15       else smulth(p, *(b->coef), a);
16     }
17     else smulth(p, *(a->coef), b);
18   }
19   else if (p->nv == 2) { // 2-variate polynomials
20     double *aa, *bb, *pp, *pp0, *af, *bf;
21     af= a->coef+a->orden; bf= b->coef+b->orden;
22     for (aa= a->coef, pp0= p->coef; aa<= af; aa++, pp0++)
23       for (bb= b->coef, pp= pp0; bb<= bf; *pp+= *aa * *bb, bb++, pp++);
24   }
25   else *(p->coef)+= *(a->coef) * *(b->coef); // 1-variate polynomials
26 }
```

Benchmarks

Estimating the overhead

An efficient implementation of the (truncated) product of power series is based on an efficient indexation of the coefficients.

We define the *overhead* as the ratio of the execution time of computing the (*k*th truncated) product over the execution time of computing $\mathrm{p_d}(k)$ products and additions.

```
1  for (i= 0; i<= k; i++) {
2     for (j= 0; j<= i; j++) {
3        nif= nch(n,i-j);
4        njf= nch(n,j);
5        for (ni= 0; ni<nif; ni++) {
6           for (nj= 0; nj<njf; nj++) {
7              z+= x*y;
8           }
9        }
10    }
11 }
```

Benchmarks

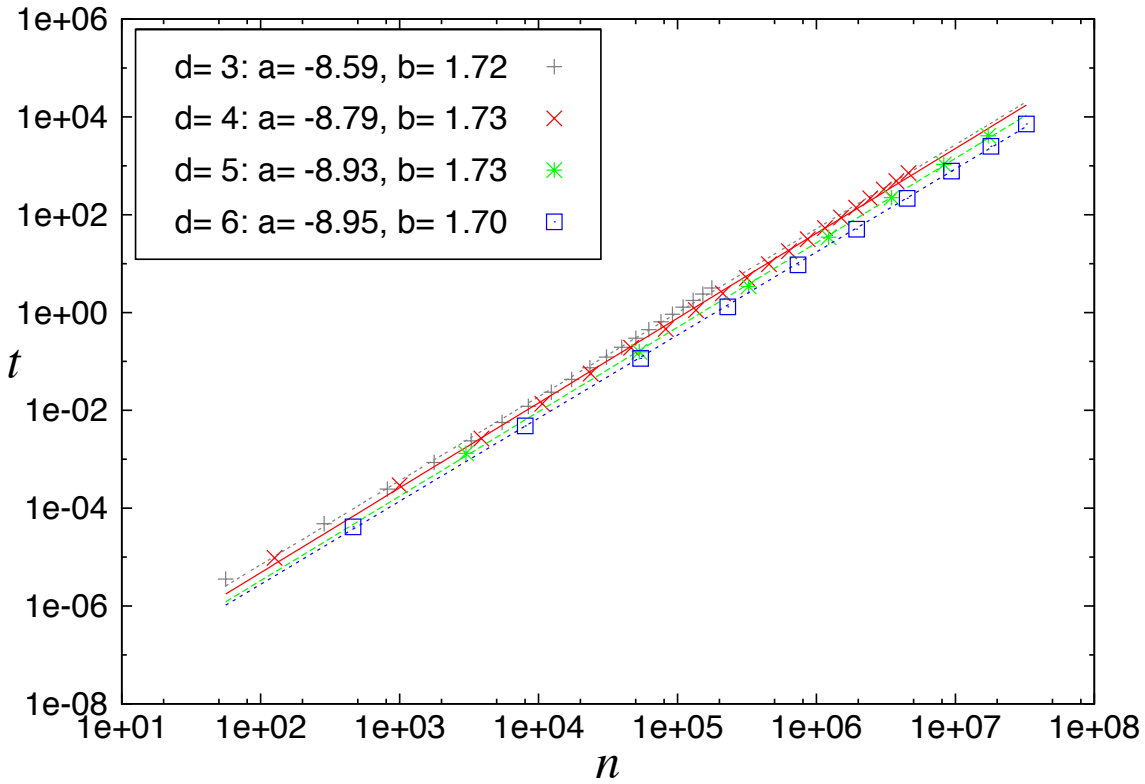
Comparison of several implementations

$d = 4$			tree			TRIP	ad hoc
k	n	p	time (s)	Mflops	over.	time (s)	time (s)
10	1001	43758	2.930e−04	149.3	3.05	NA	3.665e−04
20	10626	3108105	1.364e−02	227.9	2.50	NA	1.906e−02
30	46376	48903492	1.900e−01	257.4	2.38	2.000e−01	2.529e−01
40	135751	377348994	1.120e+00	336.9	1.70	1.630e+00	1.767e+00
50	316251	1916797311	5.050e+00	379.6	1.53	7.660e+00	1.032e+01
60	635376	7392009768	1.793e+01	412.3	1.42	2.928e+01	5.371e+01
70	1150626	23446881315	5.335e+01	439.5	1.34	9.257e+01	2.098e+02
80	1929501	64276915527	1.393e+02	461.5	1.27	2.533e+02	6.623e+02
90	3049501	157366449604	3.282e+02	479.5	1.23	6.229e+02	1.765e+03
100	4598126	352025629371	7.093e+02	496.3	1.19	1.408e+03	4.195e+03

$d = 6$			tree			TRIP	ad hoc
k	n	p	time (s)	Mflops	over.	time (s)	time (s)
10	8008	646646	4.808e−03	134.5	4.17	NA	4.451e−02
20	230230	225792840	1.310e+00	172.4	3.36	9.940e−01	1.722e+01
30	1947792	11058116888	5.043e+01	219.3	2.69	4.408e+01	9.161e+02
40	9366819	206379406870	7.808e+02	264.3	2.23	8.367e+02	1.926e+04
50	32468436	2160153123141	7.183e+03	300.7	1.96	NA	NA

Benchmarks

log-log plot of executions times



Fit $t(n) \simeq A\, n^b$, where $a = \log_{10} A$

Time is subquadratic in n (or sublinear in p)!

Benchmarks
Dependencies

- The algorithm: we use the naive formula, which involves $\mathrm{p}_d(k)$ products and additions of numbers. We take advantage of the tree data structure, and use recursivity.
- The implementation: we use the programming language C.
- The computer: iMac running under Mac OS X 10.6.4. Technical specifications: 2 GHz Intel Core Due, 4MB L2 Cache; Memory: 1 GB 667 MHz DDR2 SDRAM.
- The coefficients: these are real numbers in double-precision floating-point arithmetic, the variable type `double` in C (8 bytes per coefficient).
- The compiler: we use `gcc` with different options and flags.
- The plug: time execution can vary a lot if the laptop is not plugged in and work with the battery.

The list does not finish here.

Benchmarks
Efficient implementation

Efficient implementation!

The `Mathematica` implementation of an FFT method take hours to compute the truncated exponential up to order 10 of a 10-variate power series on a PC [Neidinger05].

Our `C` implementation of the NAIVE algorithm take less than 0.20 seconds on a slightly old laptop.

Composition with elementary functions

Make AD work!

Composition of formal power series is a very hard task, but in special cases (e.g. composing with elementary functions), the cost is proportional to that of the product!

Problem: Compute $\varphi \circ f(x)$, where φ is an elementary function (solution of a simple ode), and f is a power series.

Very well-known formulas for 1-variate power series (Knuth).

Formulas for multivariate power series?

- Reduction to 1-variate case (not on-line!).
- Use chain rule $\nabla(\varphi \circ f)(x) = \varphi'(f(x))\nabla f(x)$ [Neidinger 95, 10][Barrio 06]

Composition with elementary functions

Radial derivative

The radial derivative of a function (power series) $f(x) = \sum_{k=0}^{\infty} f_k(x)$ is defined by

$$Rf(x) = \mathrm{D}f(x)\, x = \sum_{k=0}^{\infty} k \cdot f_k(x) \; .$$

Euler Identity

For a kth order homogeneous function, f_k: $Rf_k(x) = k \cdot f_k(x)$.

Chain rule

For an univariate function $\varphi = \varphi(t)$ and a multivariate function $f = f(x)$:

$$R(\varphi \circ f)(x) = \varphi'(f(x))\; Rf(x) \; .$$

Composition with elementary functions

Applications to AD

Key observation in AD. If φ satisfies an elementary differential equation, then we can compute $\varphi \circ f$ on-line à la Knuth.

Example

If $\varphi(t) = e^t$, then $e(x) = \exp(f(x))$ satisfies $Re(x) = e(x)Rf(x)$. Since $e_0 = \exp(f_0)$, the series $e(x)$ is computed recursively by

$$e_k(x) = \frac{1}{k} \sum_{j=0}^{k-1} (k-j) f_{k-j}(x) e_j(x) \ .$$

Notice that the cost up to order k is $\sim \mathrm{p}_d(k)$.

Example

The series $p(x) = (f(x))^\alpha$ can be computed recursively from $p_0 = f_0^\alpha$ by

$$p_k(x) = \frac{1}{k\ f_0} \sum_{j=0}^{k-1} (\alpha(k-j) - j)\ f_{k-j}(x) p_j(x)\ .$$

Notice that the cost up to order k is $\sim \mathrm{p}_d(k)$.

Same tricks are applied to many others elementary functions, and the resulting algorithms have a cost that is proportional to the cost of a product. See the notes!

Comparing codes

C-XSC implementation of power function of exponent α, for 2-variate power series

```
1  ldim2taylor power(const ldim2taylor& s, const l_interval& alpha)
2  {
3   ldim2taylor erg(s.p);
4   idotprecision sum_idot;
5   l_interval sum1, sum2, h;
6
7   erg[0][0] = pow(s[0][0], alpha);
8   for(int j=1; j<=erg.p; j++) {
9       sum_idot=interval(0.0);
10      for(int i=0; i<=j-1; i++) {
11          h=alpha*(interval(j)-interval(i))-interval(i);
12          accumulate(sum_idot, h*erg[0][i], s[0][j-i]);
13        }
14      sum1 = l_interval(sum_idot);
15      erg[0][j]=sum1/interval(j)/s[0][0];
16    }
17  for(int i=1; i<=erg.p; i++) { // remaining erg(i,k)
18     for(int k=0; k<=erg.p-i; k++) {
19       sum_idot=interval(0.0);
20       for(int l=0; l<=i-1; l++) {// Do not sum coeff. (l,m)=(0,0)
21          h=alpha*(_interval(i)-_interval(l))-_interval(l);
22          for(int m=0; m<=k; m++) {
23             accumulate(sum_idot, h*erg[l][m], s[i-l][k-m]);
24           }
25         }
26       sum1 = l_interval(sum_idot);
27       sum_idot=interval(0.0);
28       for(int m=1; m<=k; m++) {
29          accumulate(sum_idot, s[0][m], erg[i][k-m]);
30         }
31       sum2 = l_interval(sum_idot);
32       erg[i][k]=(sum1/interval(i)-sum2)/s[0][0];
33      }
34    }
35  return erg;
36  }
```

Comparing codes

AD on-line implementation of power function of exponent α, for multivariate power series

```
1  void pows (serie *px, serie *x, complex alpha)
2  {
3    int k, j;
4    complex x0;
5
6    x0= coef0s(x);
7    acoef0s(px, cpow(x0,alpha));
8
9    for (k= 1; k<= px->orden; k++) {
10     zeroh(px->term[k]);
11
12     for (j= 1; j<= k; j++)
13       smprodh(px->term[k], (alpha+1)*j-k, px->term[k-j], x->term[j]);
14     multh(px->term[k], 1./(k*x0));
15   }
16 }
```

Exercise

Write a symbolic manipulator of 1-variate power series.

Exercise

Write a symbolic manipulator of 2-variate power series. Compare `C-XSC` *method, Barrio's method and on-line method.*

Elementary models

Length and complexity

Definition

A model is *elementary* if we write its equations as a finite sequence of single expressions, these meaning arithmetic operations and elementary functions such as exponential, logarithm, sinus, etc.

The *length* of the model is the number of single expressions.
The *complexity* is the sum of the complexities of the single expressions, where:

- scalar multiplication, addition, subtraction add 0;
- product, division, exponential, logarithm, power function add 1;
- square, square root add 0.5;
- trigonometric functions sin, cos, hyperbolic functions sinh, cosh add 2;
- elliptic functions sn, cn, dn add 3;
- see the notes for more examples.

Parameterizations of invariant manifolds

Computation of invariant manifolds

Setting the problem

For a vector field $\dot{z} = F(z)$ in $\mathbb{K}^n$, we assume that:

- The fixed point is the origin: $F(0) = 0$;
- The linearization has a block triangular form:

$$\mathrm{D}F(0) = \begin{pmatrix} A_1 & B \\ 0 & A_2 \end{pmatrix},$$

where A_1 is $k \times k$ and A_2 is $(n-k) \times (n-k)$.

We write $z = (\underbrace{z_1, \ldots, z_d}_{x}, \underbrace{z_{d+1}, \ldots, z_n}_{y})$.

Problem

Look for power series expansions of the parameterization of a d-dimensional invariant manifold $z = W(s)$, tangent to $y = 0$ and the corresponding dynamics $\dot{s} = f(s)$, where $s \in \mathbb{K}^d$.

Computation of invariant manifolds

The invariance equation

The invariance equation is

$$F(W(s)) = \mathrm{D}W(s)f(s)\ , \tag{2}$$

that we consider in terms of power series.
So, we write

$$\begin{aligned} x = W^x(s) &= s + \sum_{k\geq 2} W_k^x(s)\ , \\ y = W^y(s) &= \sum_{k\geq 2} W_k^y(s)\ , \end{aligned} \tag{3}$$

for the parameterization and

$$f(s) = A_1 s + \sum_{k\geq 2} f_k(s)\ , \tag{4}$$

for the reduced vector field on the manifold.

Computation of invariant manifolds

Homological equations

Step k>1

From $W_{<k}(s)$ and $f_{<k}(s)$ (and $[F(W_{<k}(s))]_{<k}$, including the intermediate results), do:

1. Compute $[F(W_{<k}(s))]_k$.
2. Solve for W_k the k-order *homological equations*

$$\mathrm{D}W_k(s)A_1 s - AW_k(s) + \mathrm{D}W_1(s)f_k(s) = R_k(s) \tag{5}$$

   where $\hat{W}_k$ is known from previous steps:

$$\hat{W}_k(s) = [F(W_{<k}(s))]_k - [\mathrm{D}W_{<k}(s)f_{<k}(s)]_k \ . \tag{6}$$

3. Compute $[F(W_{\leq k}(s))]_{\leq k}$ just adding $AW_k(s)$ to $[F(W_{<k}(s))]_k$.

Computation of invariant manifolds

Solving the homological equations

We split the homological equations in two blocks:

$$\begin{aligned} \mathrm{D}W_k^x(s)A_1 s - A_1 W_k^x(s) + f_k(s) &= R_k^x(s) + BW_k^y(s)\,, \qquad (7)\\ \mathrm{D}W_k^y(s)A_1 s - A_2 W_k^y(s) &= R_k^y(s)\,, \qquad (8) \end{aligned}$$

where $R_k(s)$ denotes the right hand side of (5).

For the sake of simplicity, we assume A_1, A_2 in diagonal form:

$$A_1 = \mathrm{diag}(\lambda_1, \ldots, \lambda_d), A_2 = \mathrm{diag}(\lambda_{d+1}, \ldots, \lambda_n).$$

Computation of invariant manifolds

Solving the homological equations

Equation (8) is written, for $i = d+1, \ldots, n$:

$$\lambda_1 \frac{\partial W_k^i}{\partial s_1} s_1 + \cdots + \lambda_d \frac{\partial W_k^i}{\partial s_d} s_d - \lambda_i W_k^i(s) = R_k^i(s) \, .$$

In particular, for $i = d+1, \ldots, n$, $|\mathbf{m}| = k$:

$$(\lambda^x \cdot \mathbf{m} - \lambda_i) W_{\mathbf{m}}^i = R_{\mathbf{m}}^i. \tag{9}$$

If there are not *cross resonances*, for $i = d+1, \ldots, n$, $|\mathbf{m}| = k$:

$$W_{\mathbf{m}}^i = \frac{R_{\mathbf{m}}^i}{\lambda^x \cdot \mathbf{m} - \lambda_i}.$$

(OK e.g. classical manifolds, but may also work for weak-stable (or slow) manifolds).

Computation of invariant manifolds

Solving the homological equations

Equation (7) is written, for $i = 1, \ldots, d$:

$$\lambda_1 \frac{\partial W_k^i}{\partial s_1} s_1 + \cdots + \lambda_d \frac{\partial W_k^i}{\partial s_d} s_d - \lambda_i W_k^i(s) + f_k^i(s) = \hat{R}_k^i(s) \,, \qquad (10)$$

where $\hat{R}_k^x(s) = R_k^x(s) + BW_k^y(s)$.

Hence, for $i = 1, \ldots, d$, $|\mathbf{m}| = k$:

$$(\lambda^x \cdot \mathbf{m} - \lambda_i) W_{\mathbf{m}}^i + f_{\mathbf{m}}^i = \hat{R}_{\mathbf{m}}^i. \qquad (11)$$

Computation of invariant manifolds

Solving the homological equations

For $i = 1, \ldots, d$, $|\mathbf{m}| = k$: $(\lambda^x \cdot \mathbf{m} - \lambda_i) W^i_{\mathbf{m}} + f^i_{\mathbf{m}} = \hat{R}^i_{\mathbf{m}}$.

Main styles of parameterizations

- *Graph method*. For $i = 1, \ldots, d$, $|\mathbf{m}| = k$.

$$W^i_{\mathbf{m}} = 0, f^i_{\mathbf{m}} = \hat{R}^i_{\mathbf{m}}.$$

- *Normal form method*. For $i = 1, \ldots, d$, $|m| = k$:

$$\begin{cases} f^i_{\mathbf{m}} = 0\ , \ W^i_{\mathbf{m}} = \dfrac{\hat{R}^i_{\mathbf{m}}}{\lambda^x \cdot \mathbf{m} - \lambda_i}, & \text{if } \lambda^x \cdot \mathbf{m} - \lambda_i \neq 0; \\ f^i_{\mathbf{m}} = \hat{R}^i_{\mathbf{m}}\ , \ W^i_{\mathbf{m}} = 0, & \text{if } \lambda^x \cdot \mathbf{m} - \lambda_i = 0. \end{cases}$$

($d = n$ corresponds to normal form)

Computation of invariant manifolds

Cost for elementary vector fields

Complexity

Let $\mathrm{w}_{n,d}(k)$ be the computational cost of solving (2) up to order k. Then,

$$\mathrm{w}_{n,d}(k) \sim C\mathrm{p}_d(k),$$

where the constant C depends on the dimension n and complexity c of the elementary vector field F, the dimension d of the manifold and the style of parameterization.

- Graph method:

$$\mathrm{w}_{n,d}(k) \sim c\ \mathrm{p}_d(k) + (n-d)d\ \mathrm{p}_d(k). \tag{12}$$

- Parameterization method (with polynomial normal form):

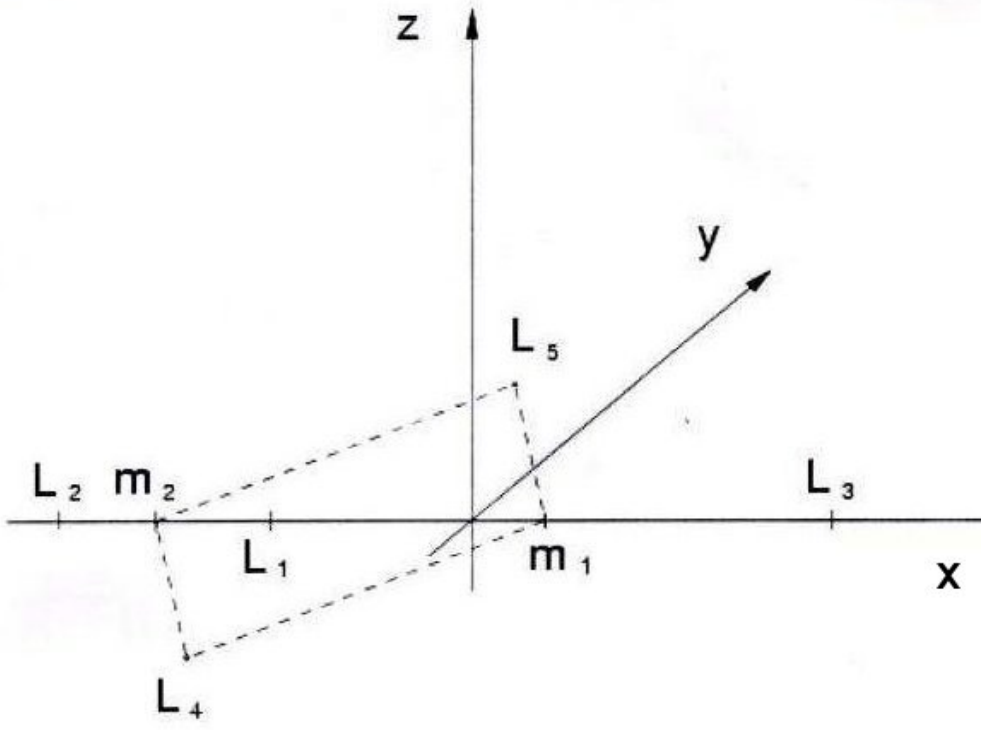
$$\mathrm{w}_{n,d}(k) \sim c\ \mathrm{p}_d(k), \tag{13}$$

An elementary example

The equations of the RTBP

Description

The RTBP models the motion of a massless body under the gravitational forces induced by two punctual bodies (primaries) in circular Keplerian motion.



The equations of the RTBP

The model

After considering a rotating frame fixing those primaries on the x-axis and z-axis perpendicular to the ecliptic plane, and doing some scalings, the equations are:

$$\begin{aligned}
\dot{x} &= p_x + y & \dot{p}_x &= p_y - \frac{1-\mu}{r_1^3}(x-\mu) - \frac{\mu}{r_2^3}(x-\mu+1) \\
\dot{y} &= p_y - x & \dot{p}_y &= -p_x - \frac{1-\mu}{r_1^3}y - \frac{\mu}{r_2^3}y \\
\dot{z} &= p_z & \dot{p}_z &= -\frac{1-\mu}{r_1^3}z - \frac{\mu}{r_2^3}z
\end{aligned} \tag{14}$$

where $r_1 = \sqrt{(x-\mu)^2 + y^2 + z^2}$, $r_2 = \sqrt{(x-\mu+1)^2 + y^2 + z^2}$, and μ is the mass parameter.
The RTBP model is elementary, with complexity 6.5.

An elementary example

Computation of center and center-(un)stable manifolds

- RTBP has 5 equilibrium points: L_1, L_2, L_3 are collinear, and L_4, L_5 are triangular.
- Each collinear point is unstable ($c \times c \times s$), with one 4D center manifold, 5D center-(un)stable manifolds, and 1D (un)stable manifolds.
- Computation of these objects is useful in astrodynamics, and has been carried out several times in the literature.
- The pioneers were Carles Simó's team in 80's, and nowadays are used e.g. in designing space missions by ESA and NASA (see Martin Lo!).
- Standard technique: reduction of the dynamics to the center manifold. This is performed through a partial normal form of the Hamiltonian killing the unstable directions (6D).
- Here: Direct computation of the center manifold of the L_1 in the Earth-Moon system.

Benchmarks

Run like hell

For $\mu \simeq 0.0121506$ (E-M system), for L_1, we have computed the center manifold ($d = 4$) with this laptop.

k	product	graph	ratio
10	4.352e–04	7.790e–03	17.90
20	2.533e–02	4.048e–01	15.98
30	3.582e–01	5.497e+00	15.34
40	2.590e+00	3.921e+01	15.14
50	1.259e+01	1.900e+02	15.09
60	4.708e+01	7.104e+02	15.08
70	1.460e+02	2.207e+03	15.12

The theoretical estimates for the ratio is $14.5 = 6.5 + (6 - 4) \cdot 4$.

Benchmarks

Comparison

In faster (but similar) machines, we have compared different algorithms.

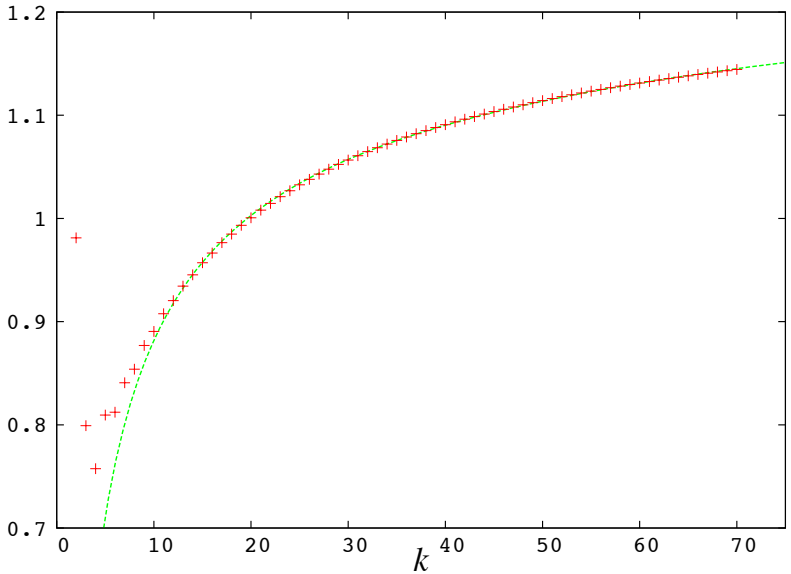
k	Lie series	Graph transform	Graph style + AD
8	0m 0.085s	0m 0.057s	0m 0.005s
16	0m 3.876s	0m 2.943s	0m 0.084s
24	2m 10.251s	1m 13.965s	0m 0.790s
32	33m 22.000s	14m 35.475s	0m 4.764s

(Results for Lie series and Graph transform from [FJ10])

Growth of the coefficients

Fitting the growth

For $k > 0$, let $\ell_1(k)$ be the maximum of the ℓ_1 norms of the kth order coefficients of W_k. In log-scale ...



Then:

$$\ell_1(k) \sim \ell(k) = A\lambda^k (\log k)^{ck}$$

where $a = \log A, b = \log \lambda$ and c are estimated by

$$a = -1.25 \pm 0.05\ ,\ b = -0.212 \pm 0.008\ ,\ c = 0.252 \pm 0.005$$

Growth of the coefficients

Mild growth implies sharp behaviour

Assume that, for δ small enough, the expansion of $W(s)$ for $|s|_\infty \leq \delta$ is asymptotic, that is:

$$|W_{\leq k}(s) - W(s)|_\infty \leq \ell_1(k+1)\delta^{k+1} \sim \varepsilon(\delta, k) = \ell(k+1)\delta^{k+1} \ . \quad (15)$$

Following [C. Simó], the best bound $b(\delta)$ for the error in the approximation of $W(s)$ by $W_{\leq k}(s)$ in the box $|s|_\infty \leq \delta$ is obtained taking $k = k(\delta)$ minimizing $\varepsilon(\delta, k)$. In the present case,

$$k(\delta) = \frac{1}{e} \exp\left((\lambda\delta)^{-\frac{1}{c}}\right) - 1 \ ,$$
$$b(\delta) = \varepsilon(\delta, k(\delta)) = A \exp\left(-c(\lambda\delta)^{\frac{1}{c}}(k(\delta)+1)\right) \ .$$

Hence, the mild growth of the coefficients of the expansions explains the behavior of the (best) error of the asymptotic approximation.

Growth of the coefficients

Mild growth implies sharp behaviour

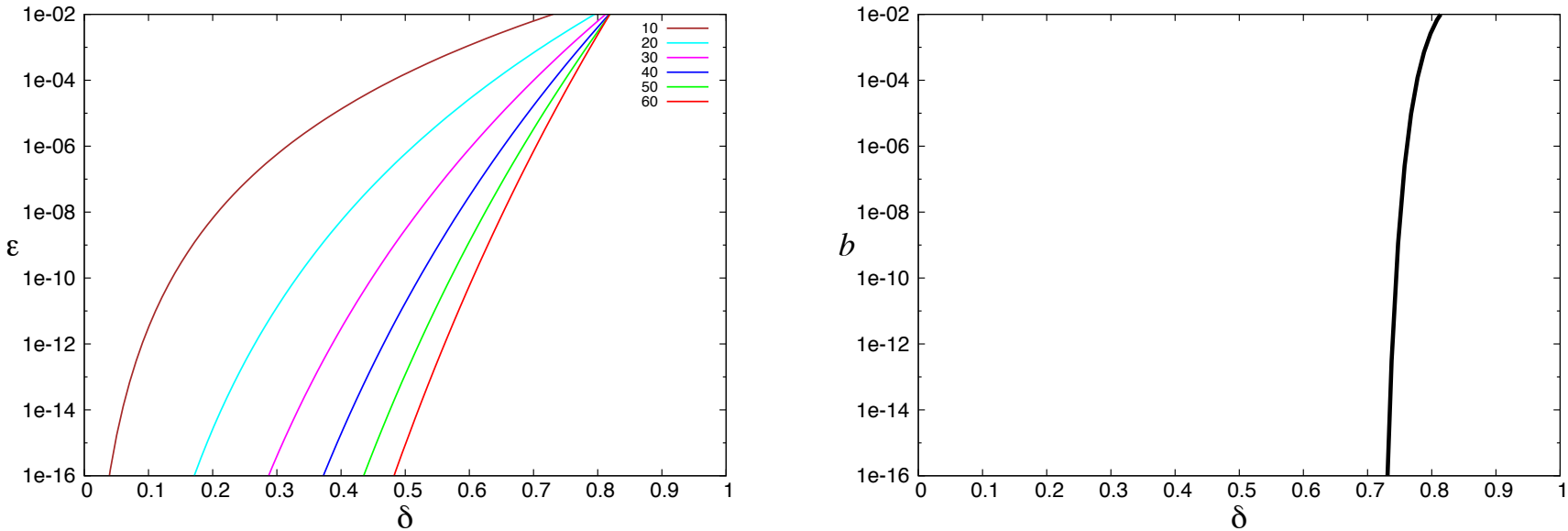


Figure: Error estimates as a function of δ for different order of approximations (left), and best error estimates (in log-scale).

Dynamics on the center manifold

Poincaré section

In order to analyze the dynamics on the 4D center manifold, we use the following standard technique:

- Fix an energy level $H > H_{L_1} \simeq -1.594171$;
- Use Poincaré section with $\{z = 0\}$.

From different energy levels one gets a collection of 2D phase portraits, obtaining a local-global view of the dynamics on the center manifold.

The boundary of the intersection of the center manifold with an energy level and the Poincaré section is a closed curve: a planar Lyapunov orbit.

Other important periodic orbits are the vertical Lyapunov orbit and the halo orbits.

Dynamics on the center manifold

Computation of Poincaré maps

Two methods of integration of orbits:

- Reduction: Integrate numerically the vector field on the manifold, using e.g. a R-K 7-8 with automatic stepsize control.
  This is quite reliable but numerically expensive.
- Projection: Given a point on the section, integrate the *full* vector field, using e.g. a Taylor method of order 18 with automatic step size control. At each return map, project the point on the center manifold.
  This is numerical cheap.

Dynamics on the center manifold

Error estimates

- The error in the invariance equation:

$$e_I(t, s_0) = \|F(W(s(t))) - \mathrm{D}W(s(t))f(s(t))\|_\infty \ ,$$

- The error in the orbit:

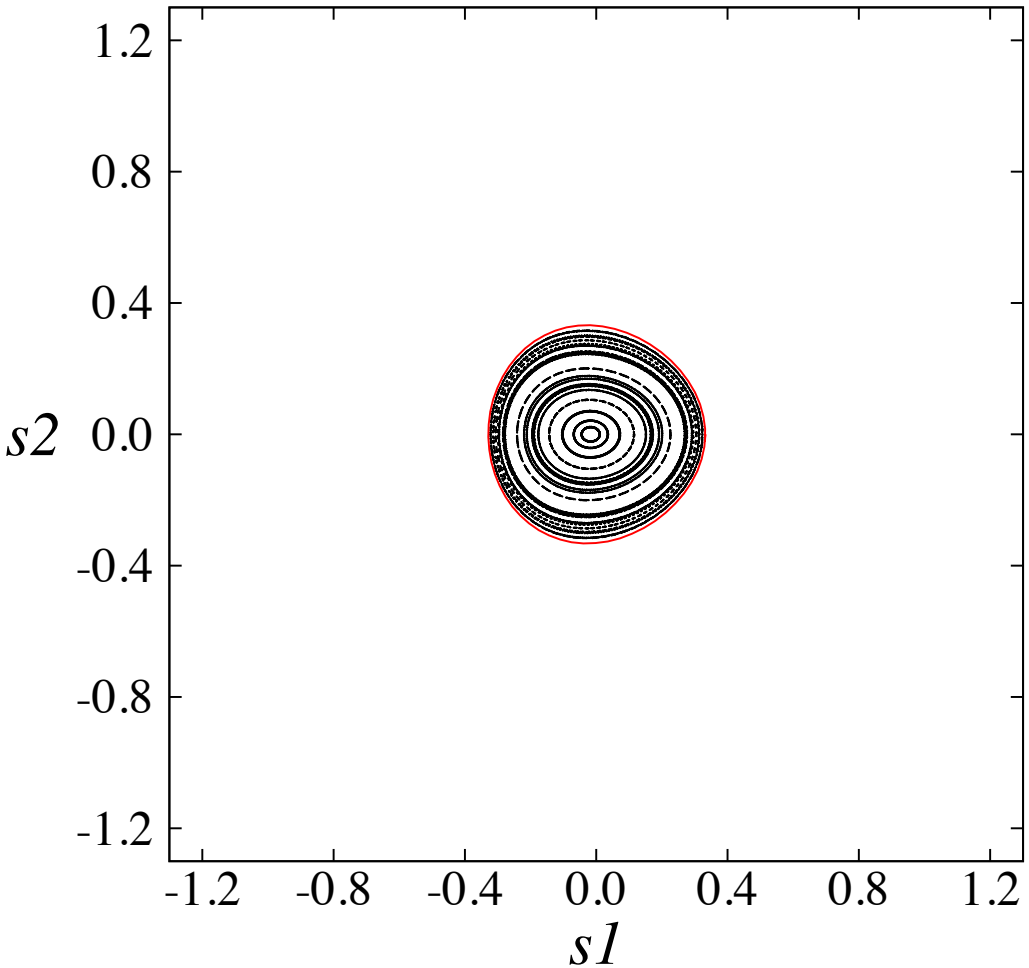
$$e_O(t, s_0) = \|W(s(t)) - z(t)\|_\infty .$$

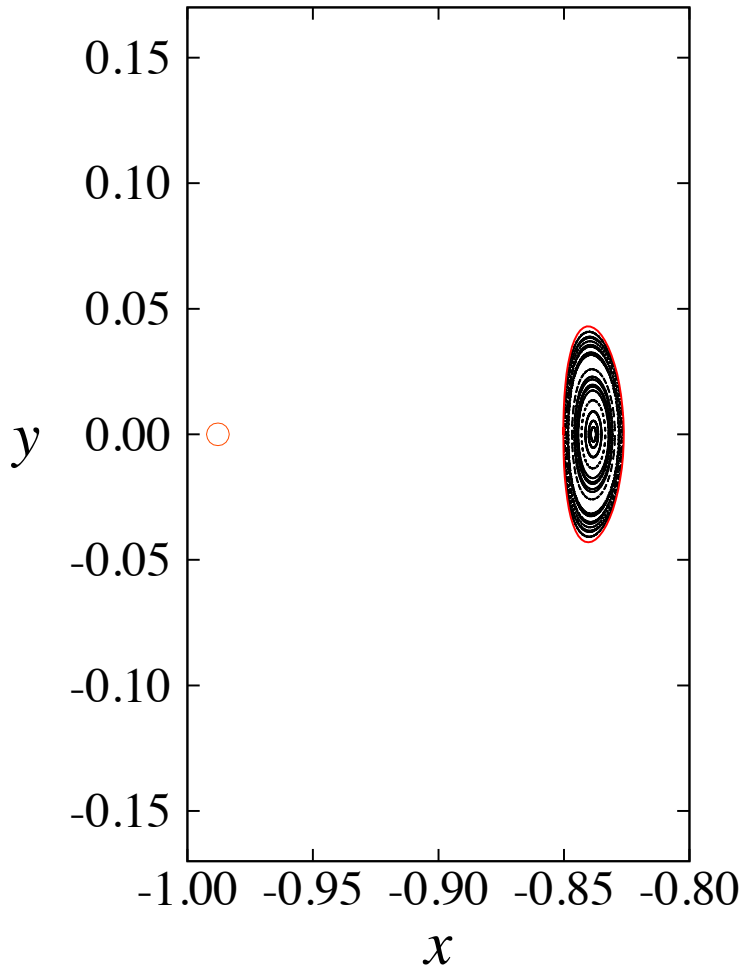
- The error in the Hamiltonian:

$$e_H(t, s_0) = |H(W(s(t))) - H(W(s(0)))|.$$

Dynamics on the center manifold

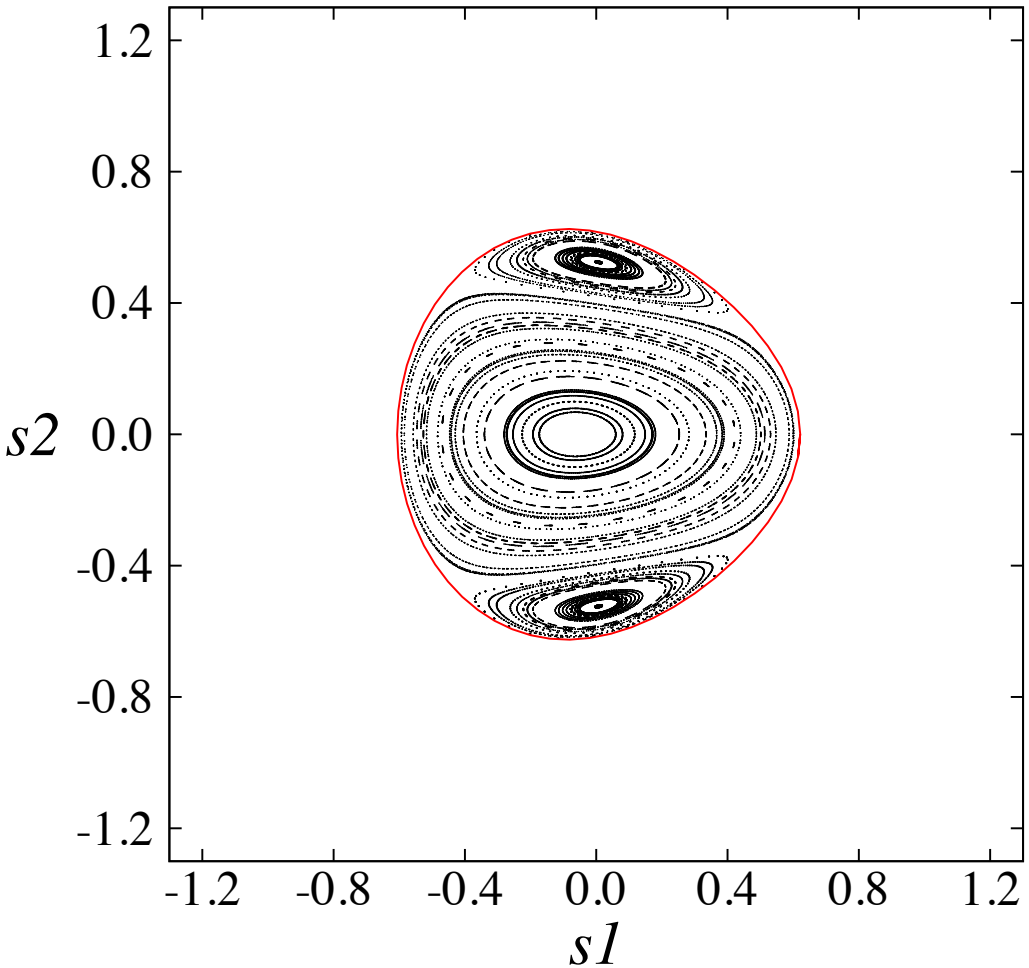
$H = -1.590$

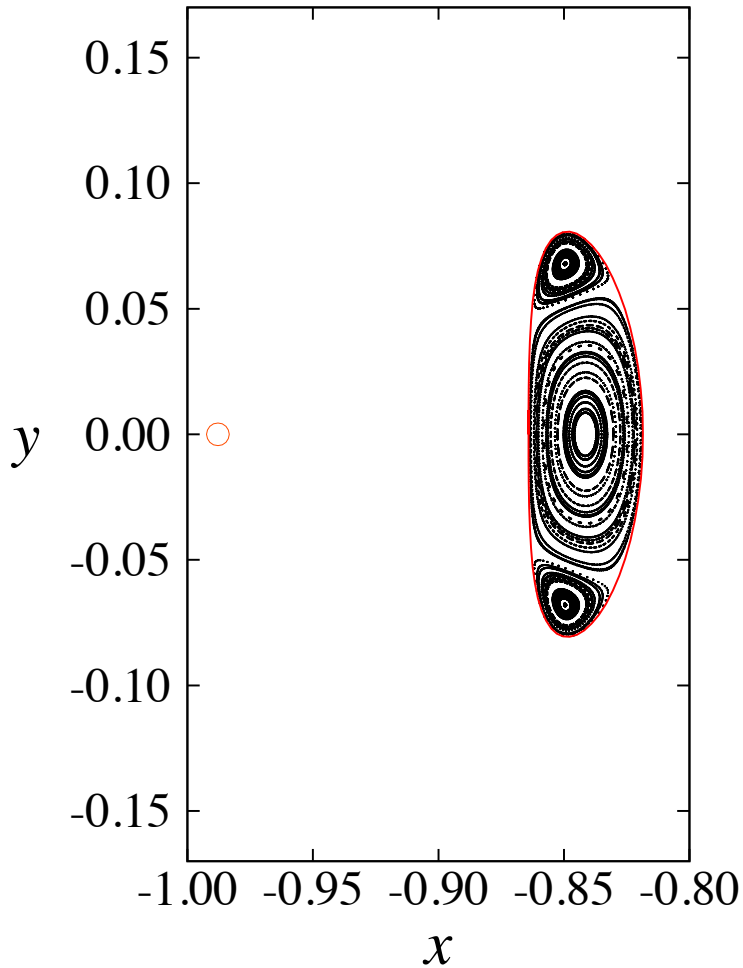




Dynamics on the center manifold

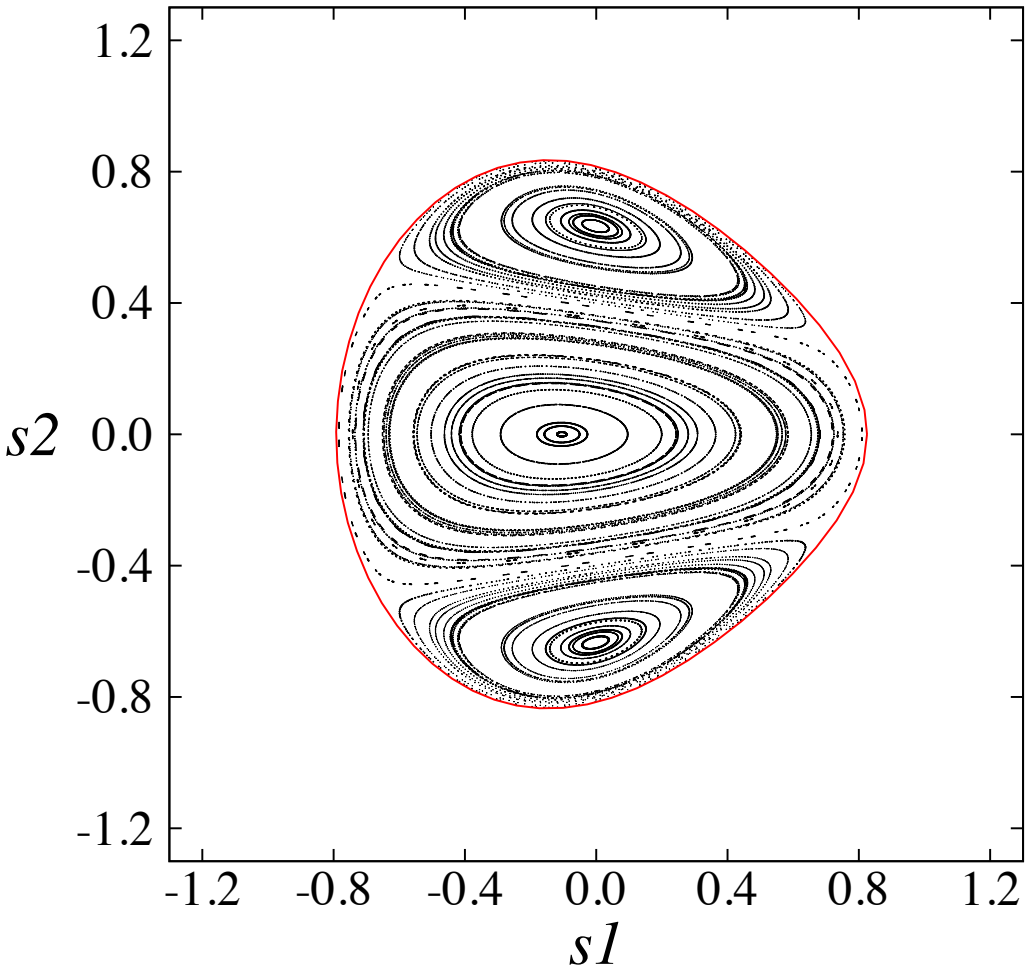
$H = -1.580$

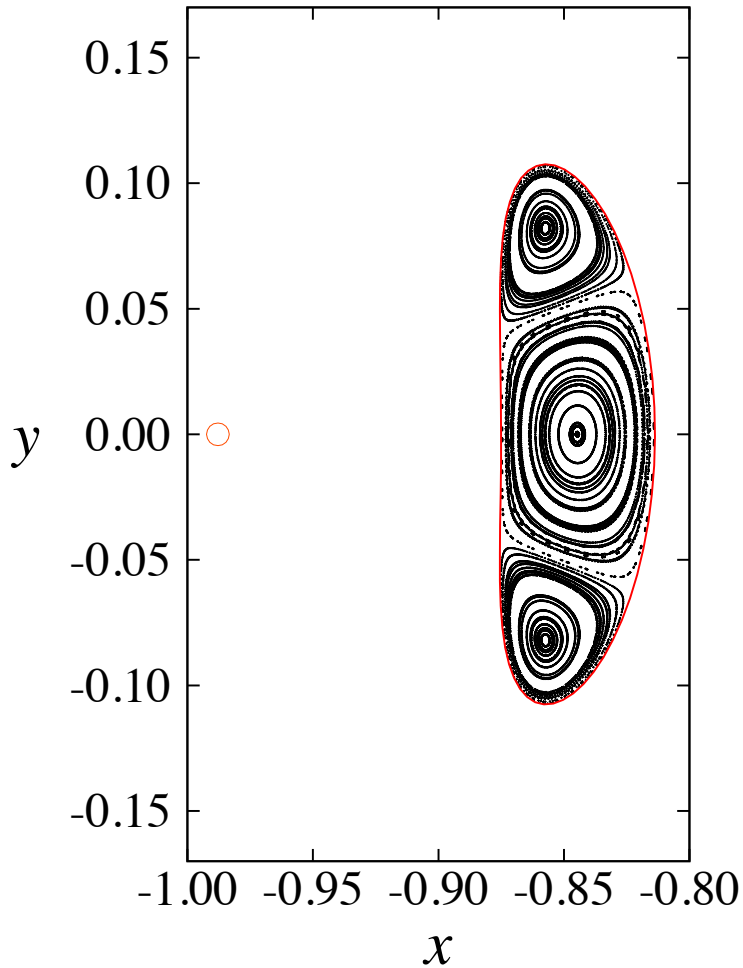




Dynamics on the center manifold

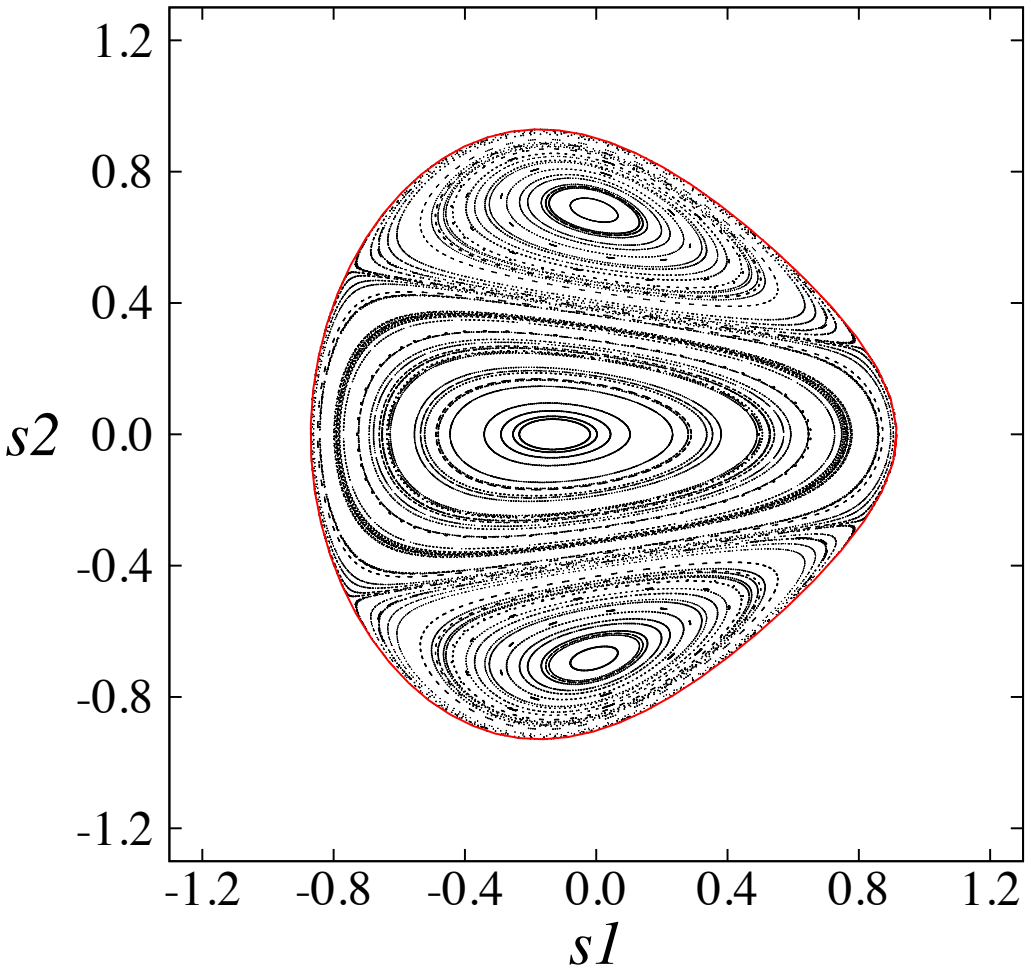
$H = -1.570$

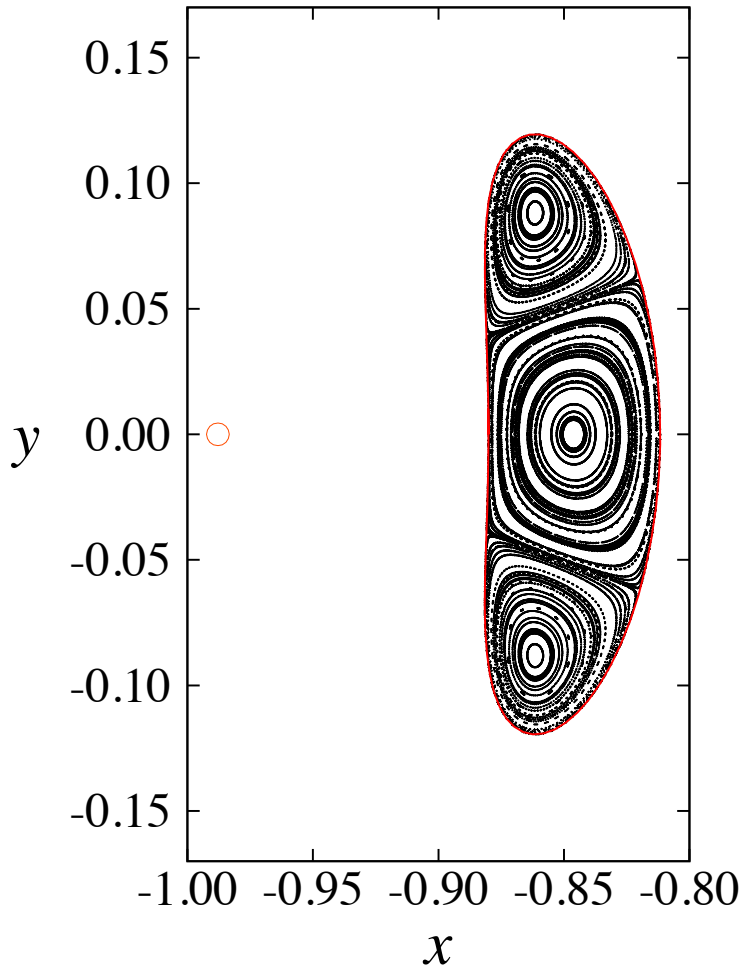




Dynamics on the center manifold

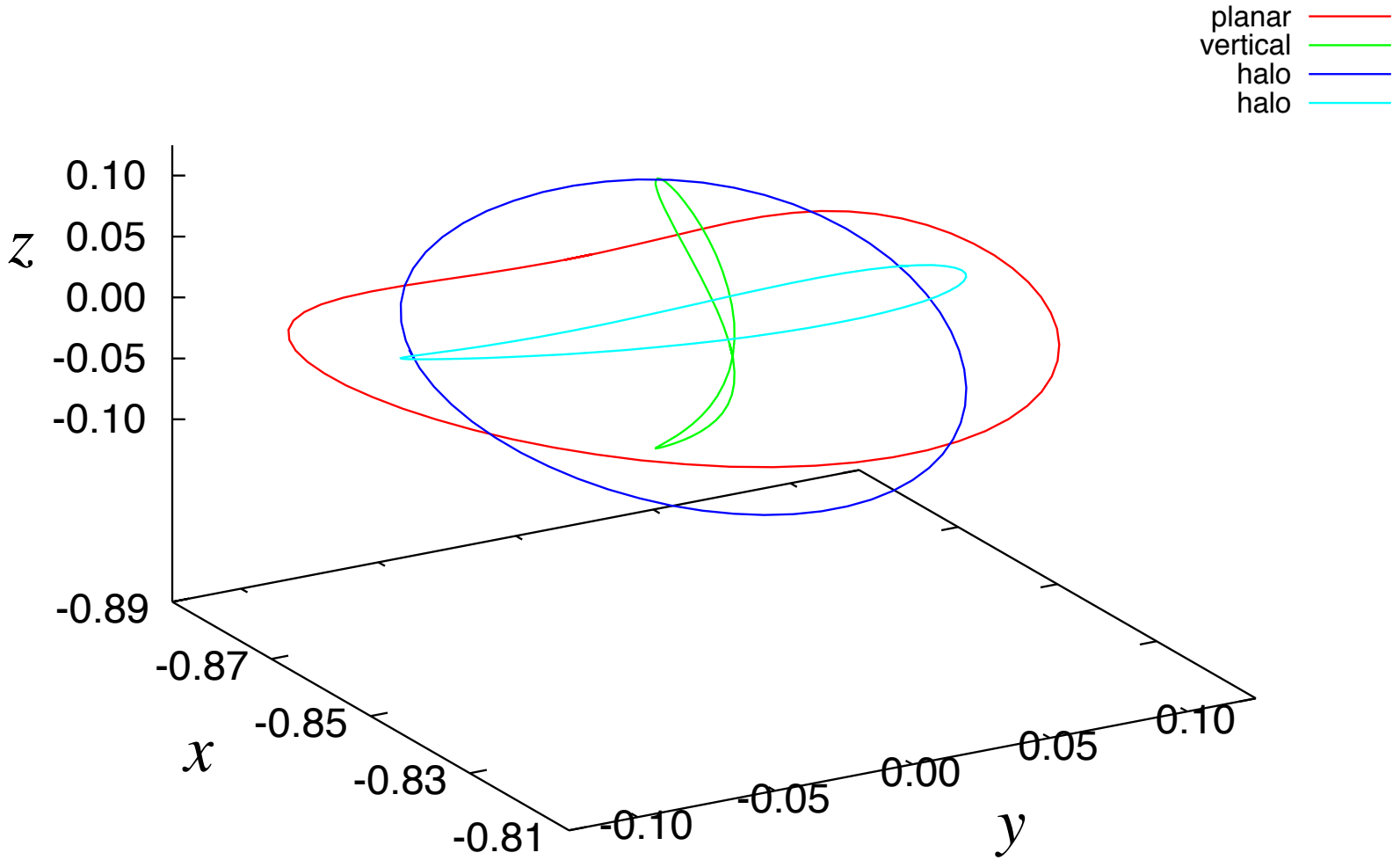
$H = -1.565$





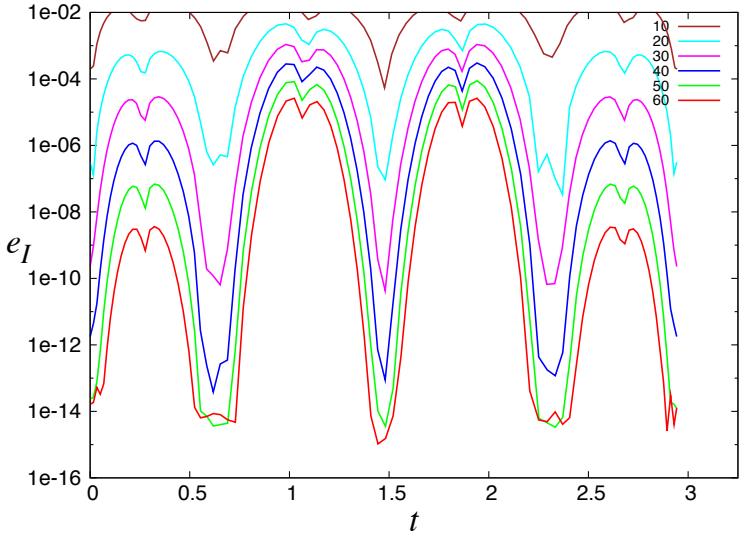
The energy level $H = -1.565$

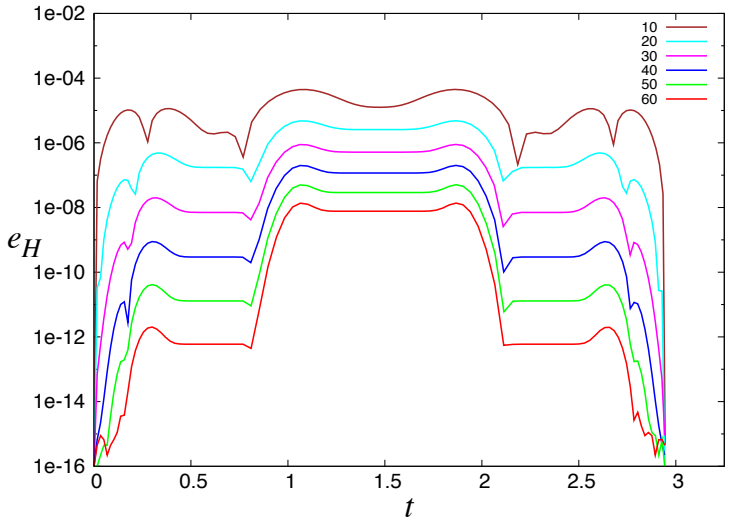
Main periodic orbits

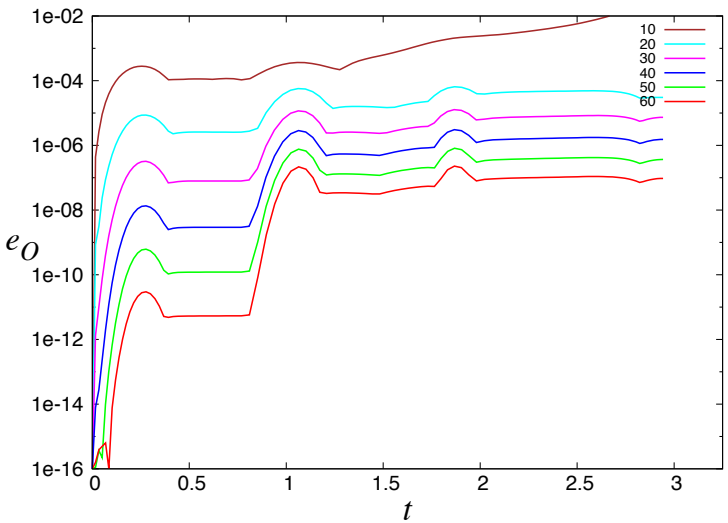


The energy level $H = -1.565$

Error estimates for the planar Lyapunov orbit



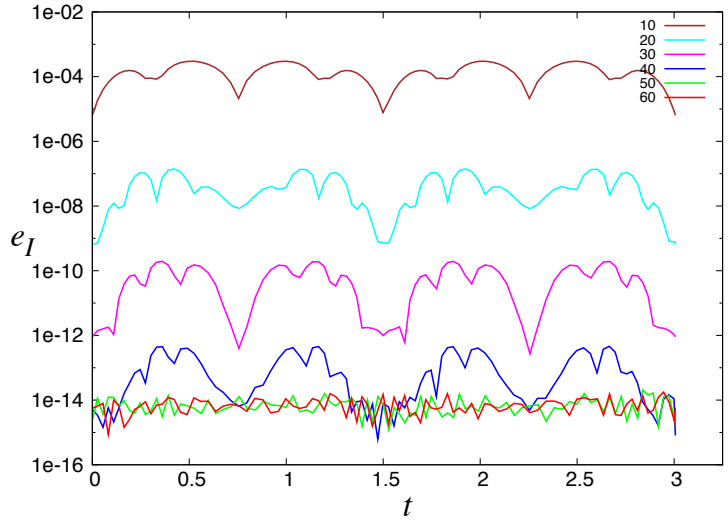


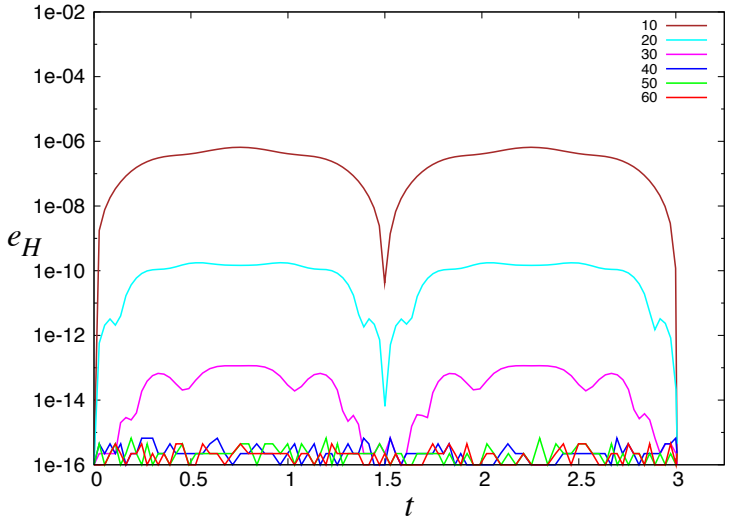


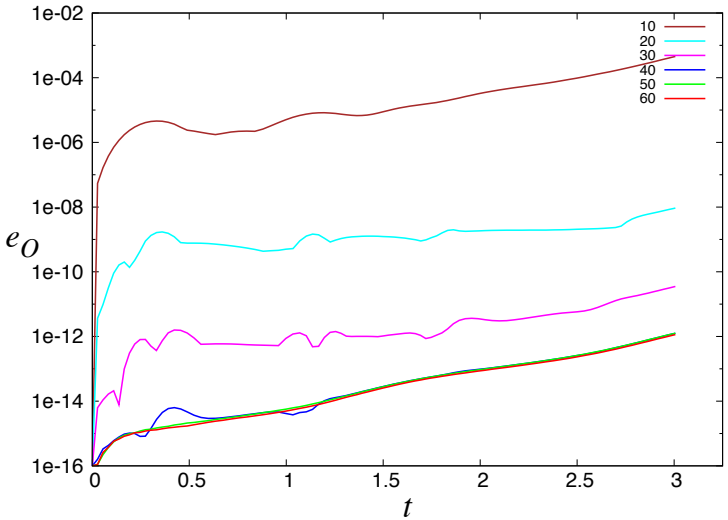
k	T	e^1_{PO}	e^2_{PO}
10	2.944806	2.48e-15	3.35e-02
20	2.943184	9.99e-16	3.03e-05
30	2.943083	1.67e-15	7.38e-06
40	2.943068	1.44e-15	1.53e-06
50	2.943065	1.67e-15	3.67e-07
60	2.943065	1.44e-15	9.53e-08

The energy level $H = -1.565$

Error estimates for the vertical Lyapunov orbit



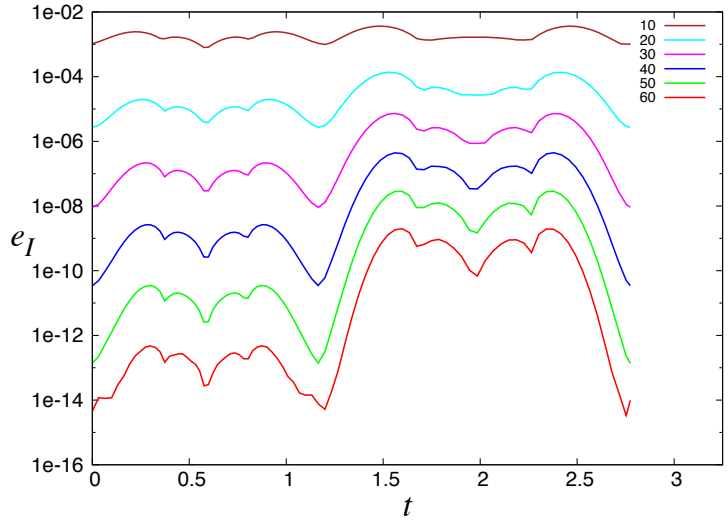


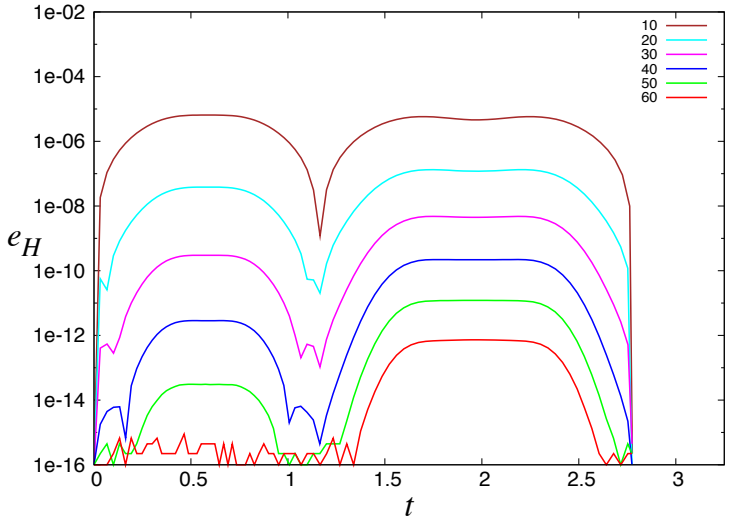


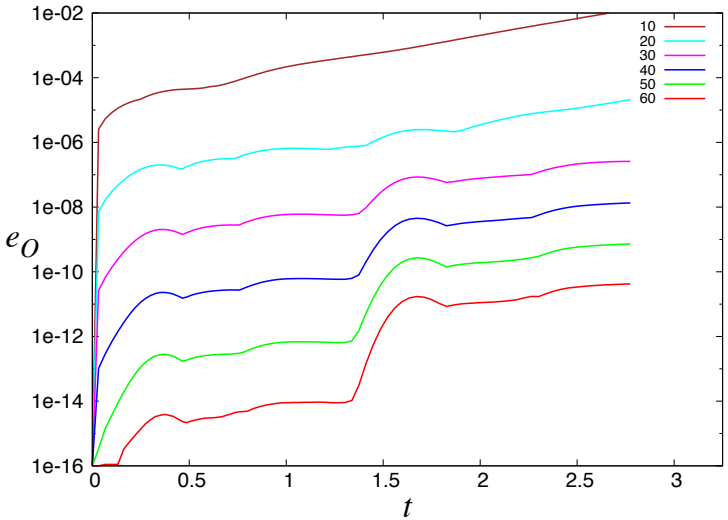
k	T	e^1_{PO}	e^2_{PO}
10	3.006224	9.99e-16	4.54e-04
20	3.006243	8.88e-16	9.33e-09
30	3.006243	7.77e-16	3.51e-11
40	3.006243	3.89e-16	1.25e-12
50	3.006243	4.37e-16	1.27e-12
60	3.006243	4.37e-16	1.14e-12

The energy level $H = -1.565$

Error estimates for the halo orbit



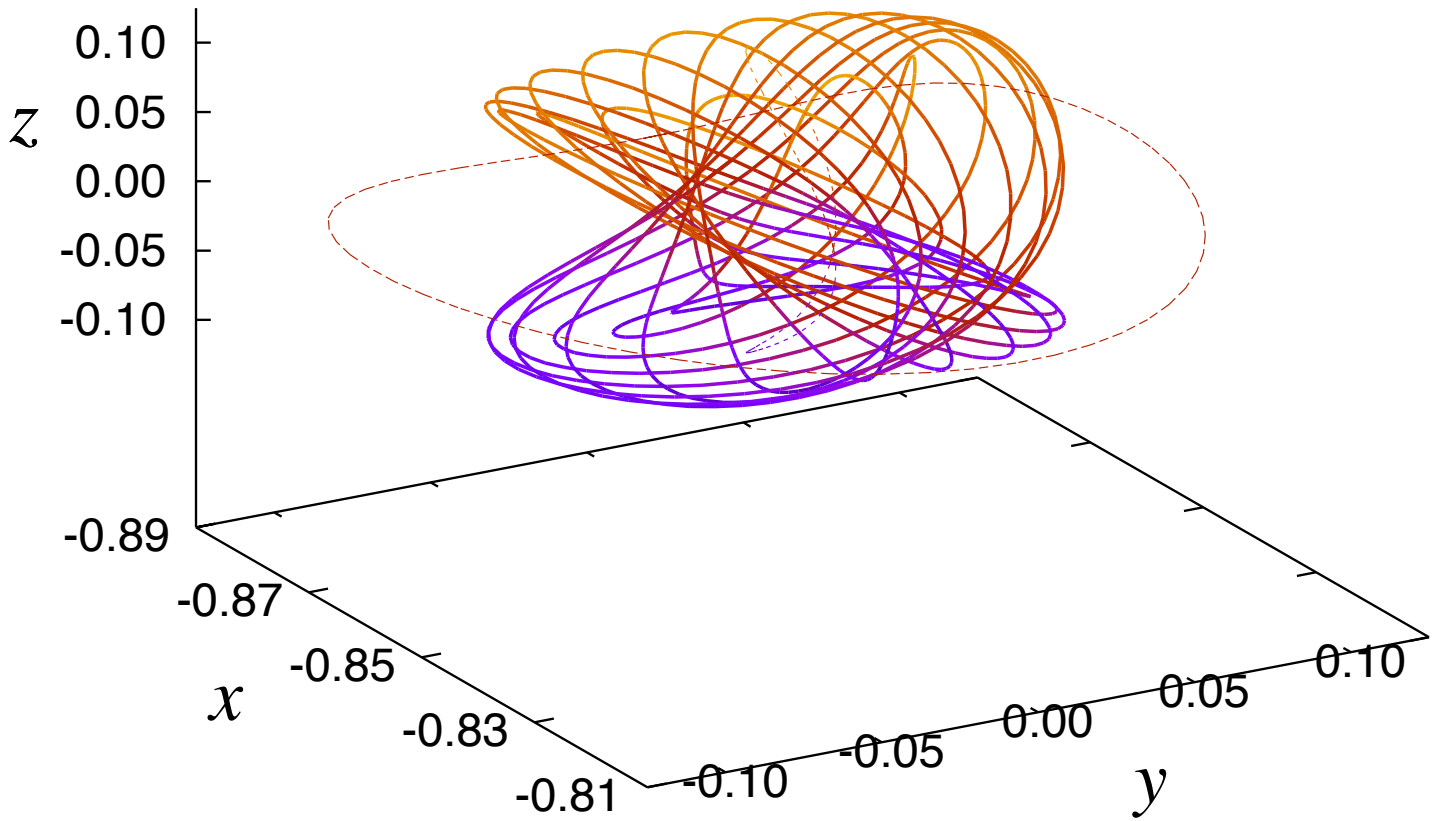




k	T	e^1_{PO}	e^2_{PO}
10	2.775478	1.22e-15	1.31e-02
20	2.774291	2.78e-16	2.11e-05
30	2.774274	2.54e-14	2.59e-07
40	2.774274	2.66e-15	1.34e-08
50	2.774274	1.33e-15	7.24e-10
60	2.774274	7.77e-16	4.23e-11

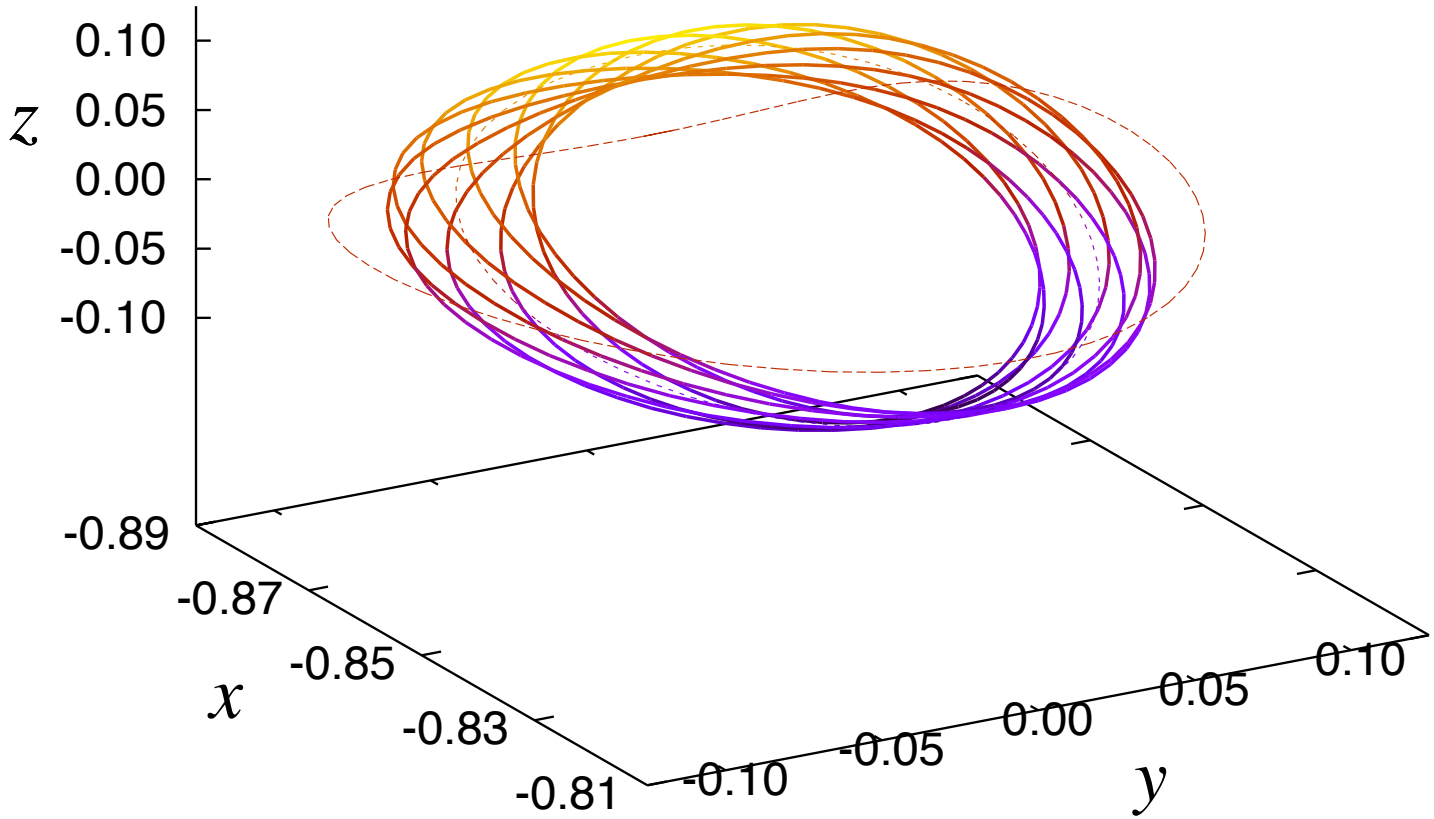
The energy level $H = -1.565$

A -1 : 18 period orbit around vertical orbit



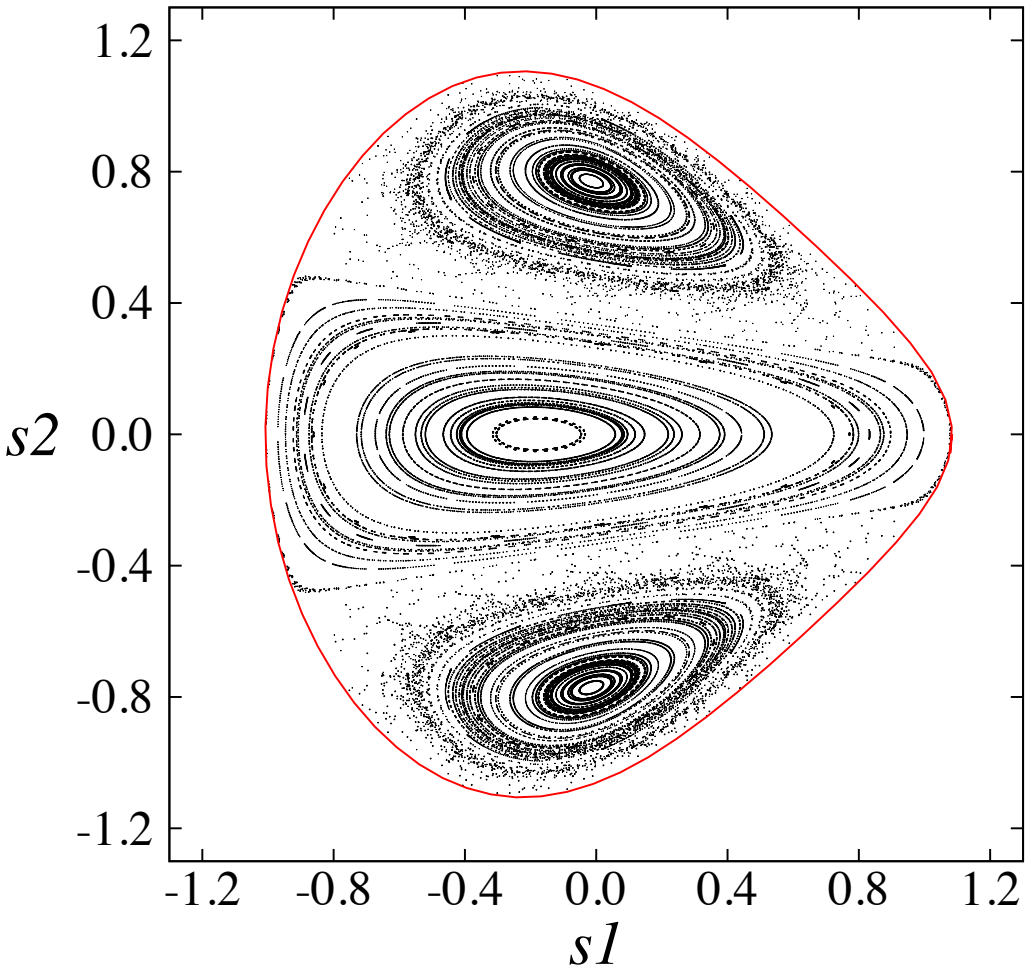
The energy level $H = -1.565$

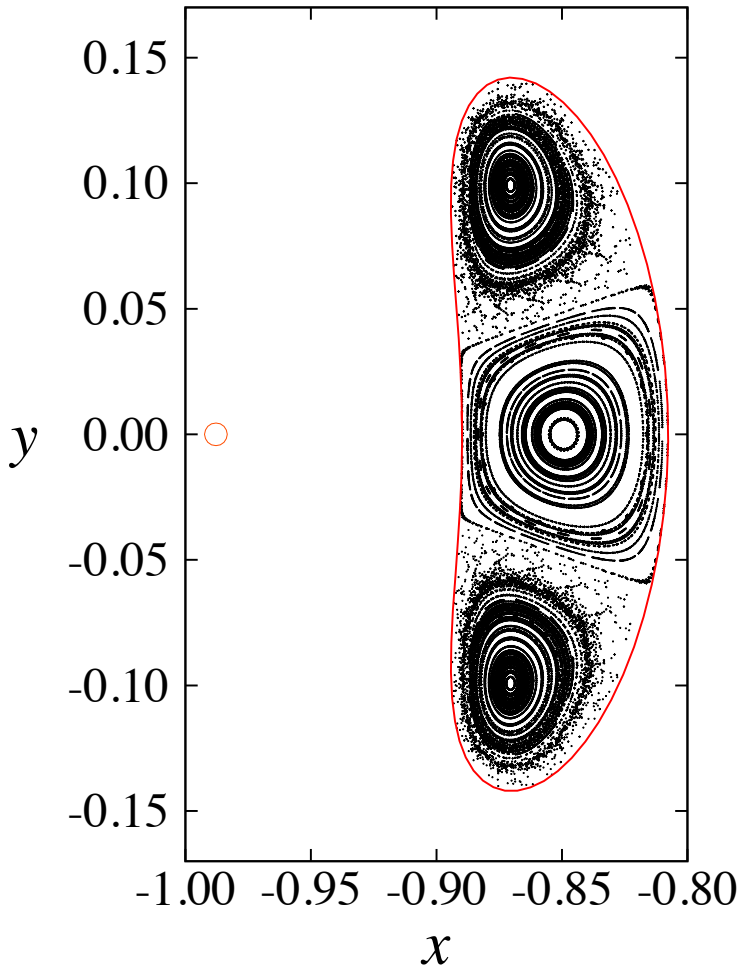
A 1 : 9 periodic orbit around halo orbit



Further on

Numerical chaos in the energy level $H = -1.555$





Conclusions

Conclusions

- The link of the parameterization method with automatic differentiation provide efficient methods to compute multivariate power series expansions of invariant manifolds.
- Computation of the semi-local manifold is the starting point to study the manifold (globalization, visualization, etc.)
- The methods work for conservative and dissipative systems.
- We can also obtain efficient methods to compute normal forms of elementary Hamiltonians.

Final remarks

AD methods are applied in many contexts:

- Numerical methods
- Validated methods of computation (with interval arithmetics)
- Sensitivity analysis
- Design optimization
- Data assimilation and inverse problems