

# Recent Advances in the Rigorous Integration of Flows of ODEs with Taylor Models

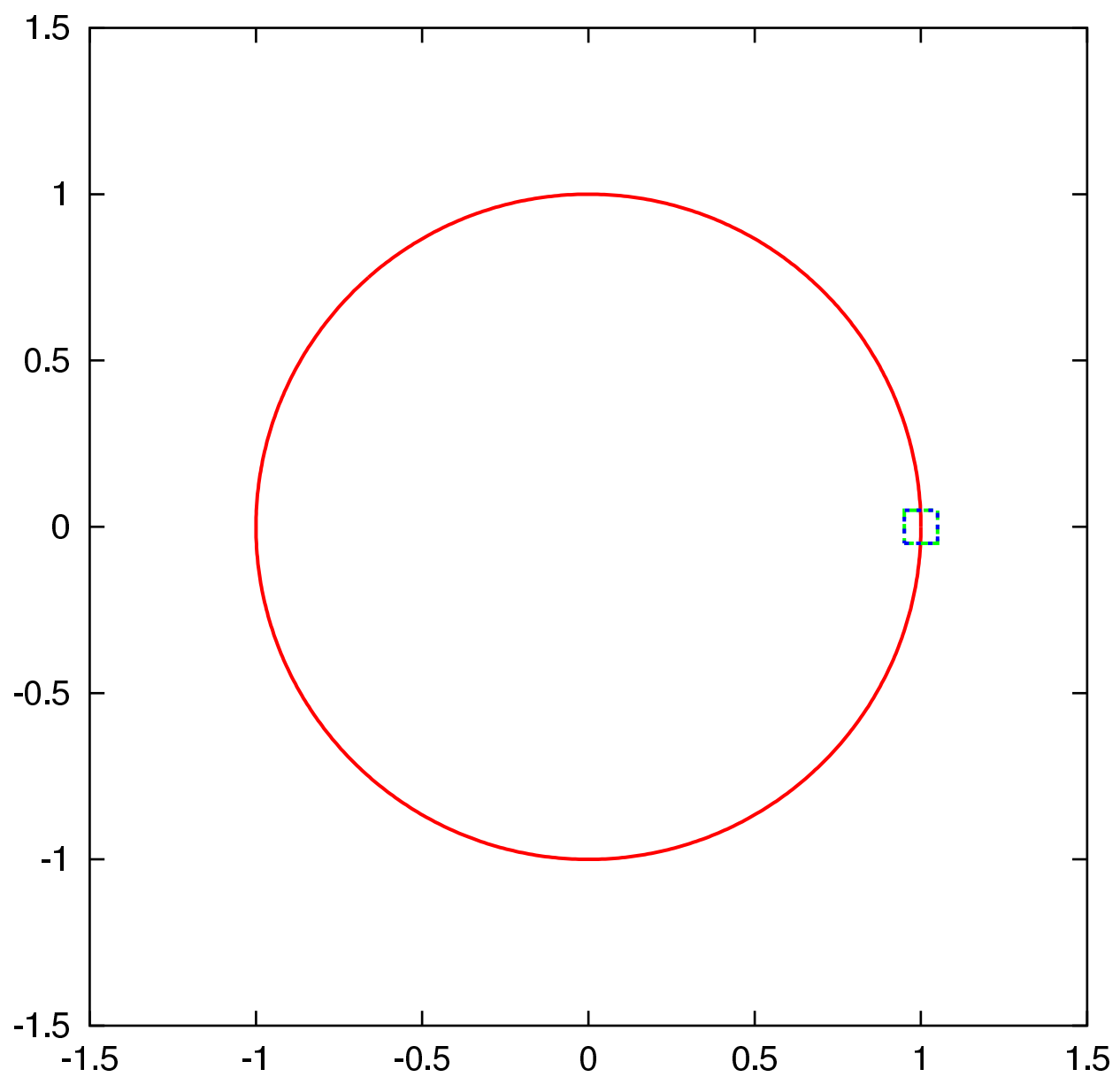
Kyoko Makino and Martin Berz

Department of Physics and Astronomy  
Michigan State University

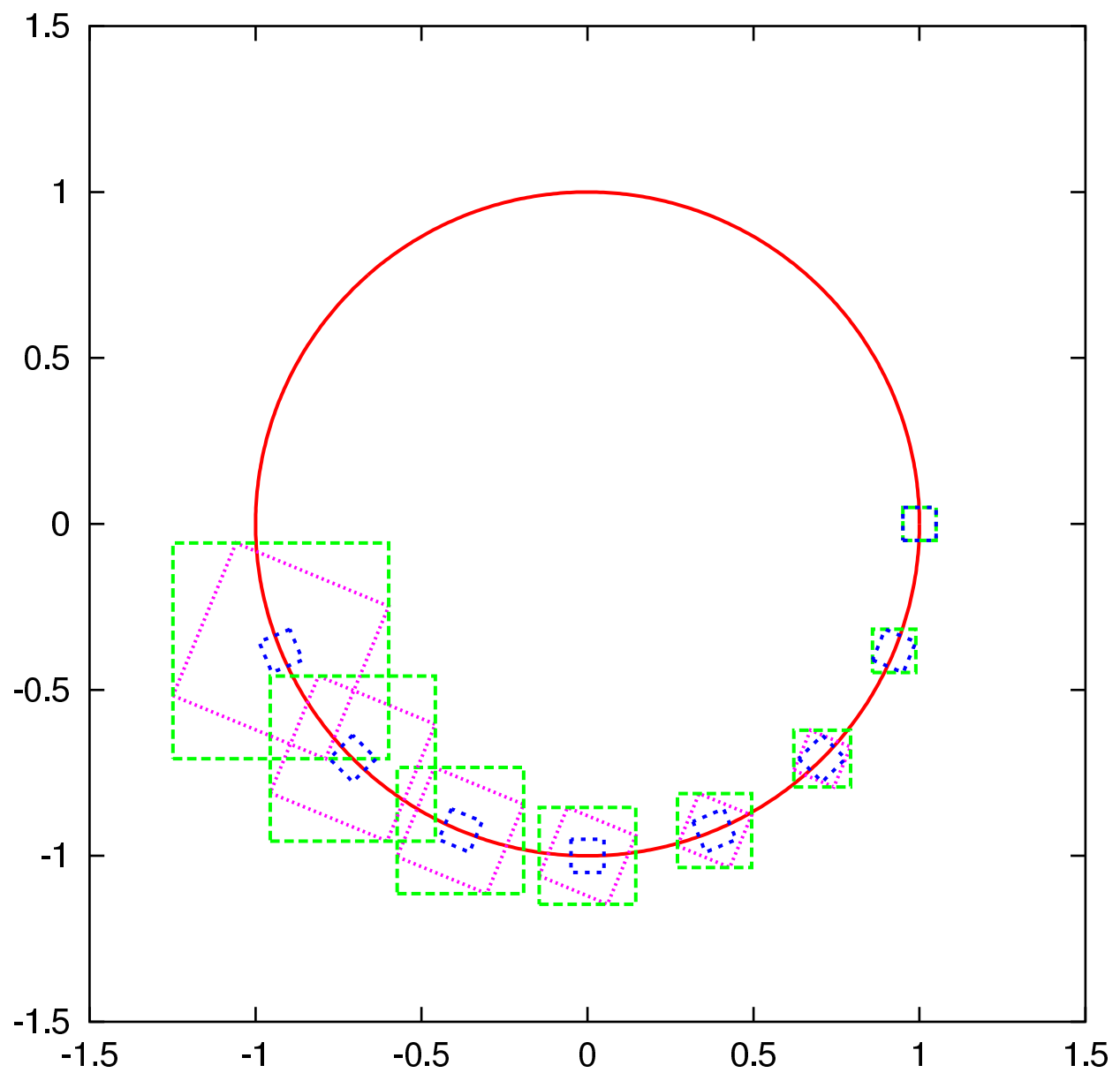
# Outline

1. Review of the old version of COSY-VI
2. The Reference Trajectory and the Flow Operator
3. Step Size Control
4. Error Parametrization of Taylor Models
5. Dynamic Domain Decomposition
6. Examples

To transport a large phase space volume with validation,



**Over Estimation has to be controlled.**





# Review of the Old Version of COSY-VI

Version 2 (2004)

# Key Features and Algorithms of COSY-VI

- High order expansion not only in time  $t$  but also in transversal variables  $\vec{x}$ .
- Capability of weighted order computation, allowing to suppress the expansion order in transversal variables  $\vec{x}$ .
- Shrink wrapping algorithm including blunting to control ill-conditioned cases.
- Pre-conditioning algorithms based on the Curvilinear, QR decomposition, and blunting pre-conditioners.
- Resulting data is available in various levels including graphics output.

# The Volterra Equation

Describe dynamics of two conflicting populations

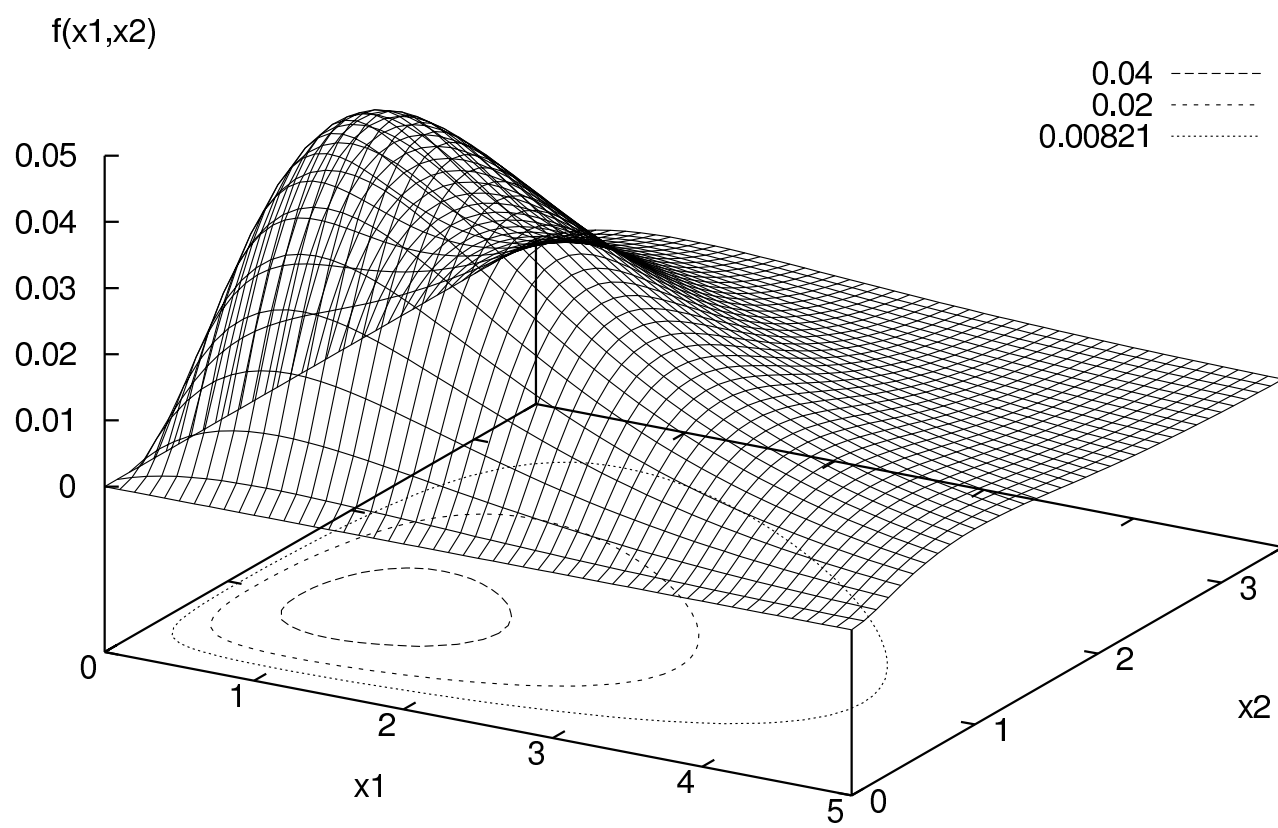
$$\frac{dx_1}{dt} = 2x_1(1 - x_2), \quad \frac{dx_2}{dt} = -x_2(1 - x_1)$$

Interested in initial condition

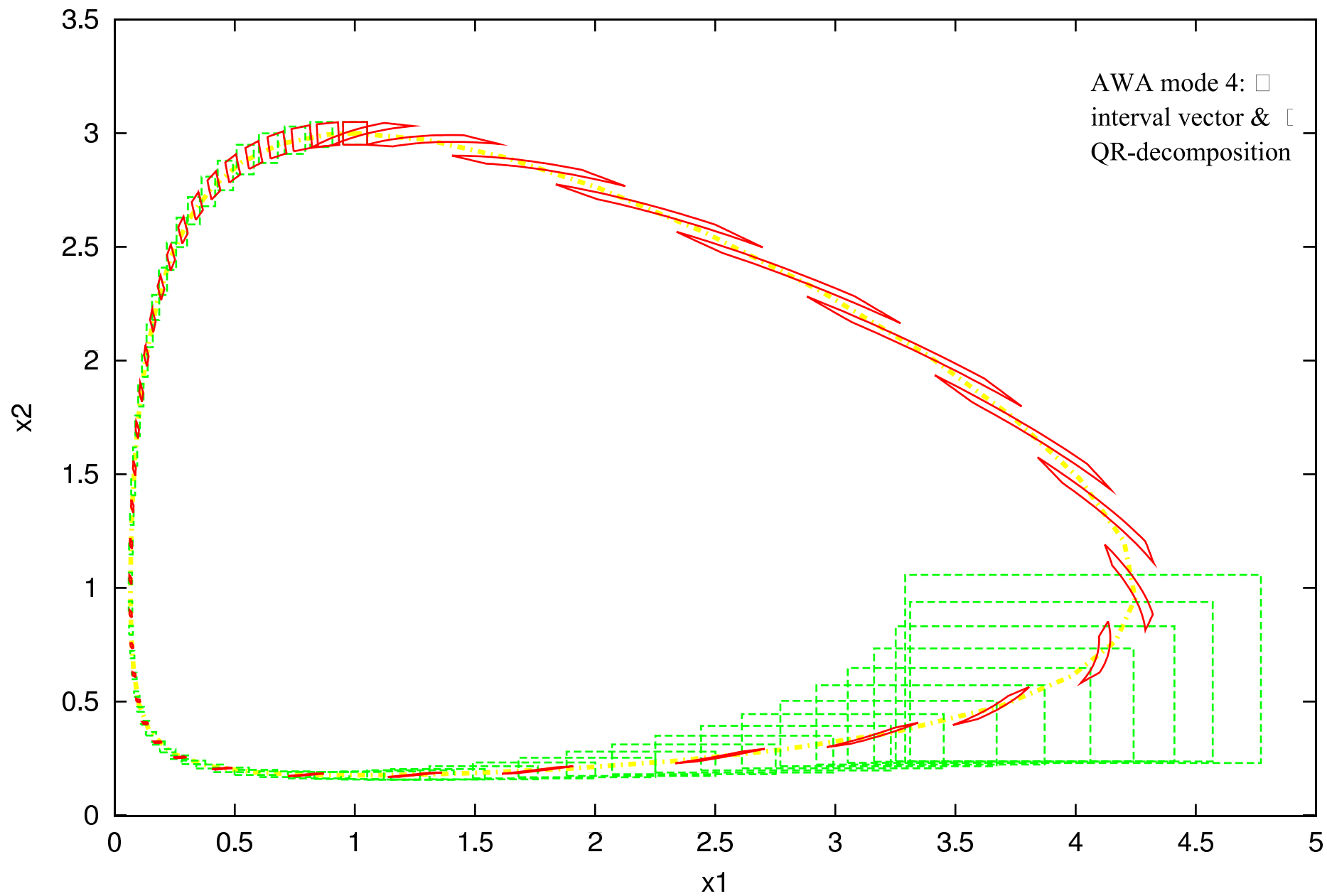
$$x_{01} \in 1 + [-0.05, 0.05], \quad x_{02} \in 3 + [-0.05, 0.05] \quad \text{at } t = 0.$$

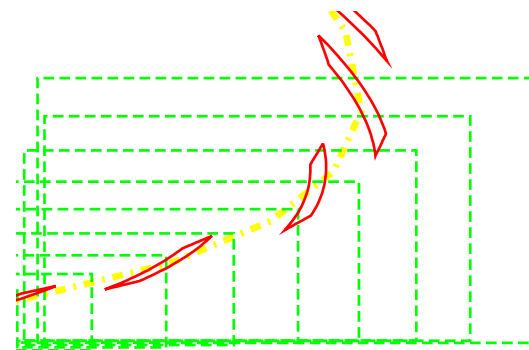
Satisfies constraint condition

$$C(x_1, x_2) = x_1 x_2^2 e^{-x_1 - 2x_2} = \text{Constant}$$



# Integration of the Volterra eq. COSY-VI and AWA





## 2 Rössler equations

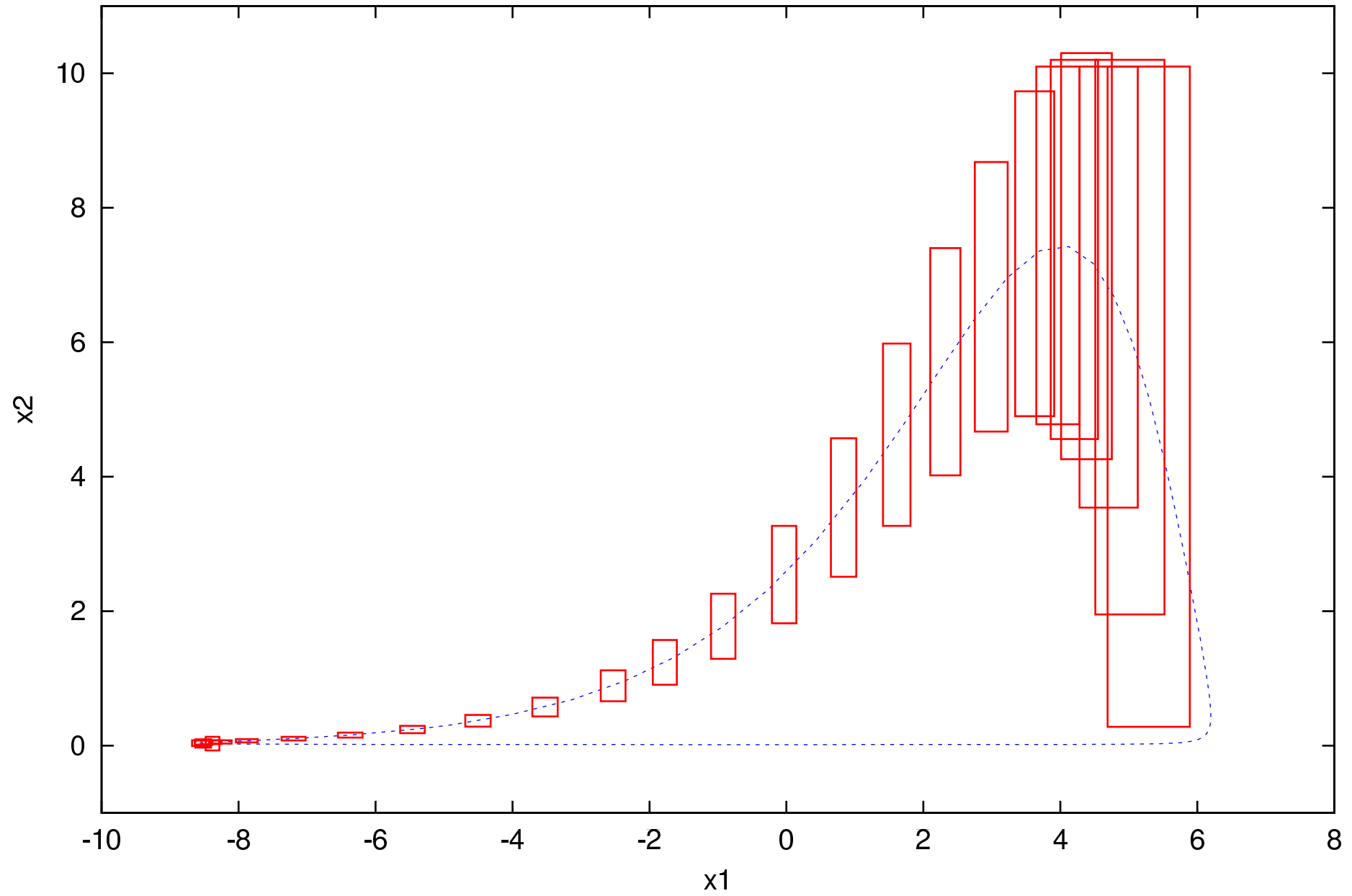
The Rössler equations are given by

$$\begin{aligned}x' &= -(y + z) \\y' &= x + 0.2y \\z' &= 0.2 + z(x - a),\end{aligned}\tag{4}$$

where  $a$  is a real parameter. We focus here at the value of  $a = 5.7$ , where numerical simulations suggest an existence of a strange attractor.

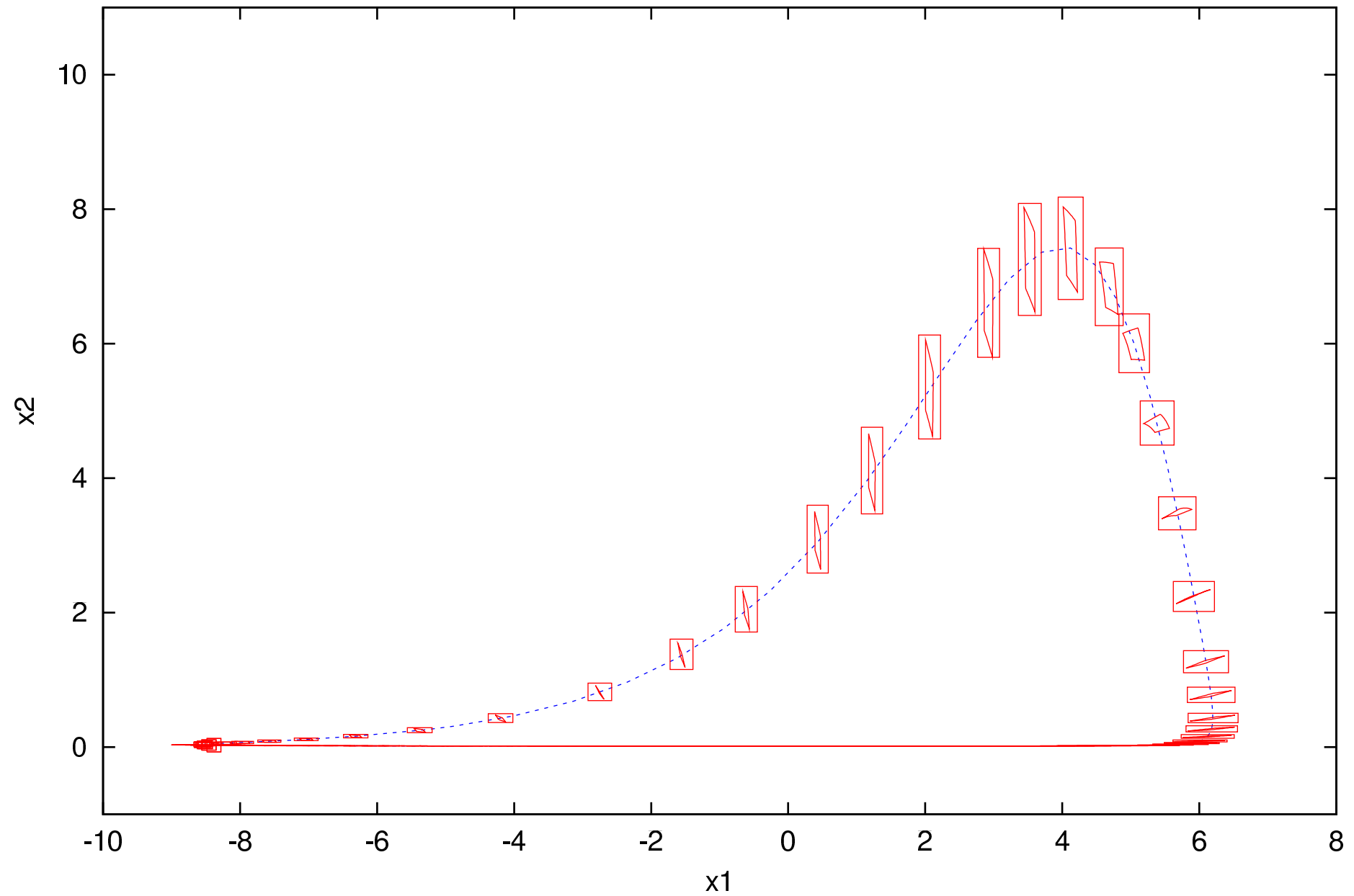
On section  $x = 0$  we consider the following initial condition  $(y, z) \in (-8.38095, 0.0295902) + [-\delta, \delta]^2$ , where  $\delta$  should be considerably larger than  $10^{-3}$ . The integration time should be around  $T = 6$ .

# AWA Integration of the Roessler eqs.

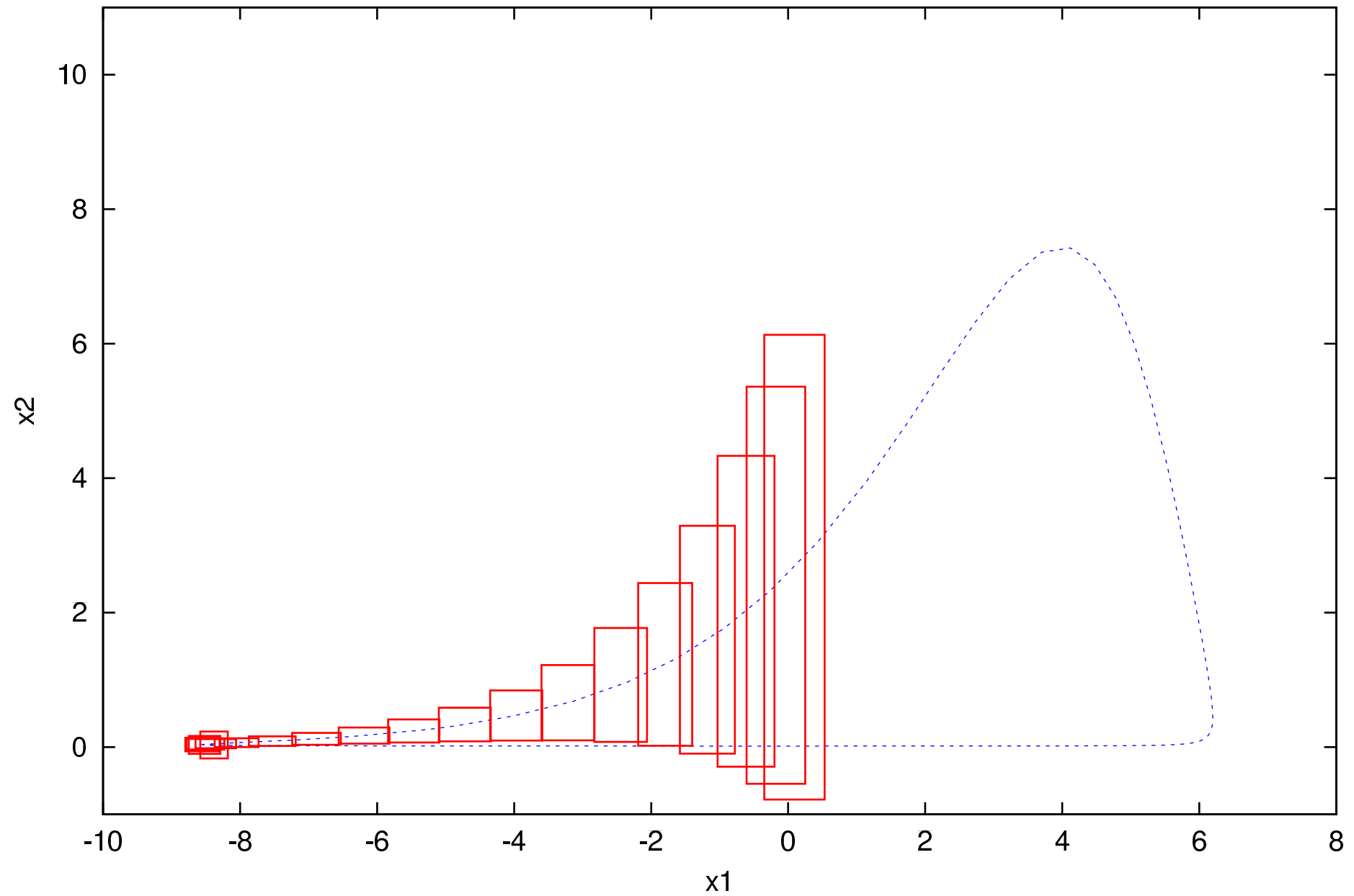




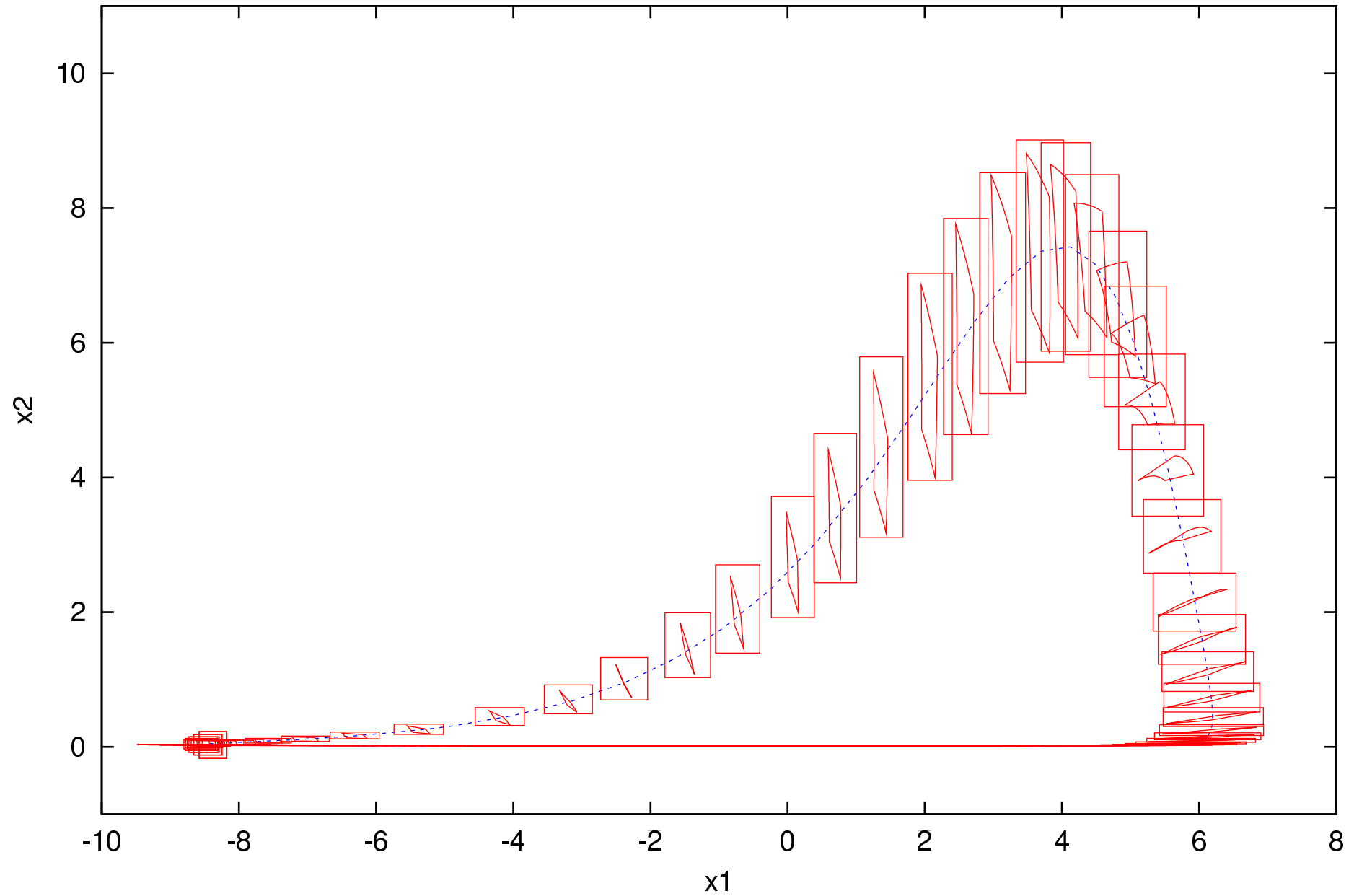
# COSY-VI Integration of the Roessler eqs.

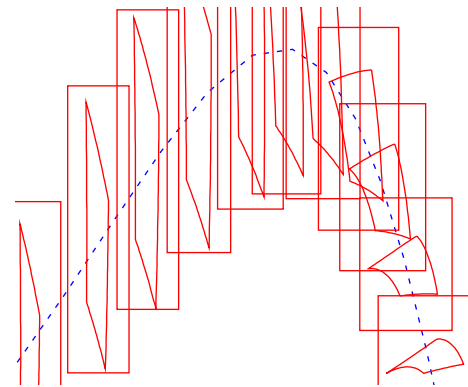


# AWA Integration of the Roessler eqs.



# COSY-VI Integration of the Roessler eqs.





# The Henon Map

Henon Map: frequently used elementary example that exhibits many of the well-known effects of nonlinear dynamics, including chaos, periodic fixed points, islands and symplectic motion. The dynamics is two-dimensional, and given by

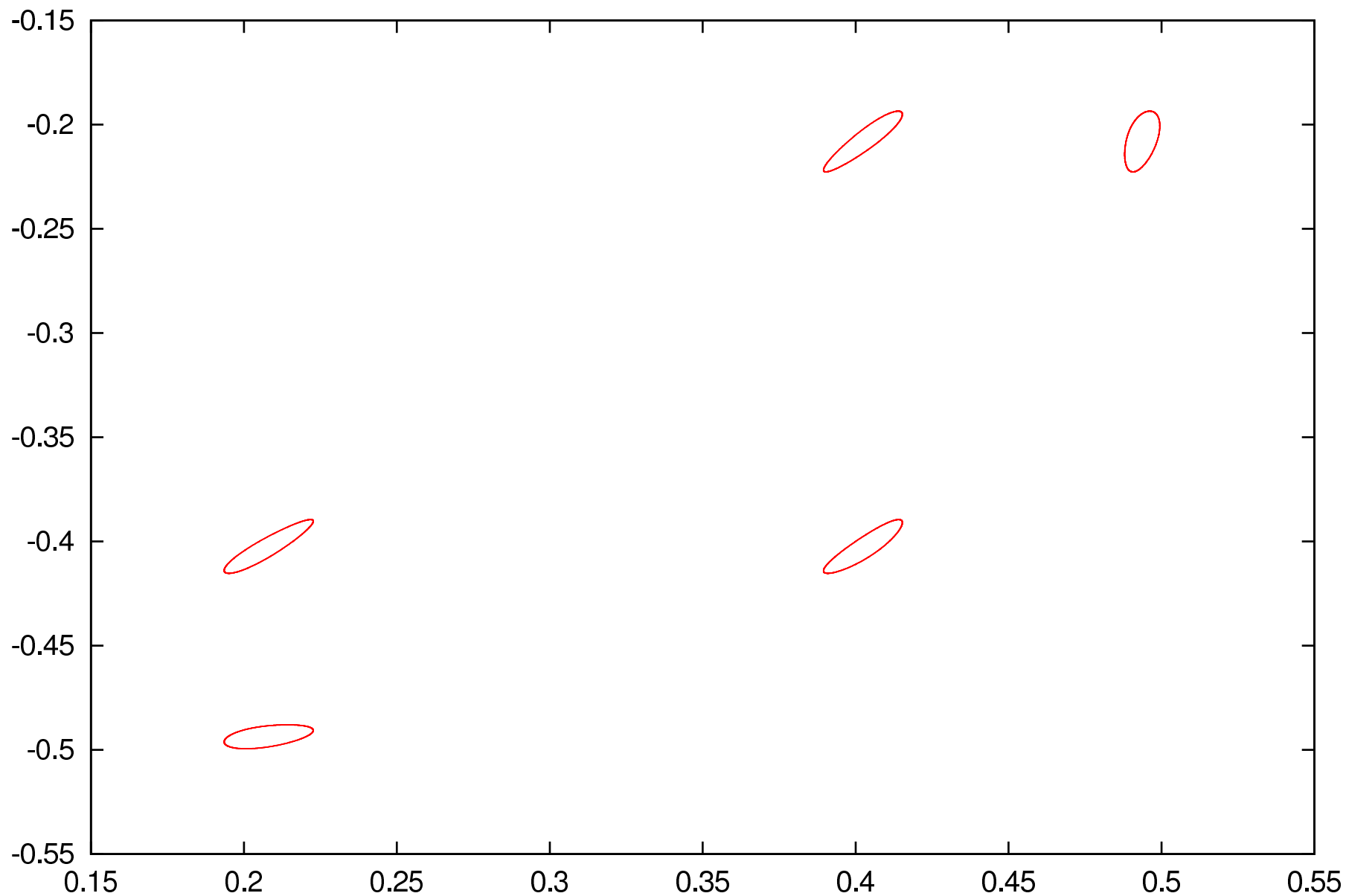
$$\begin{aligned}x_{n+1} &= 1 - \alpha x_n^2 + y_n \\ y_{n+1} &= \beta x_n.\end{aligned}$$

It can easily be seen that the motion is area preserving for  $|\beta| = 1$ . We consider

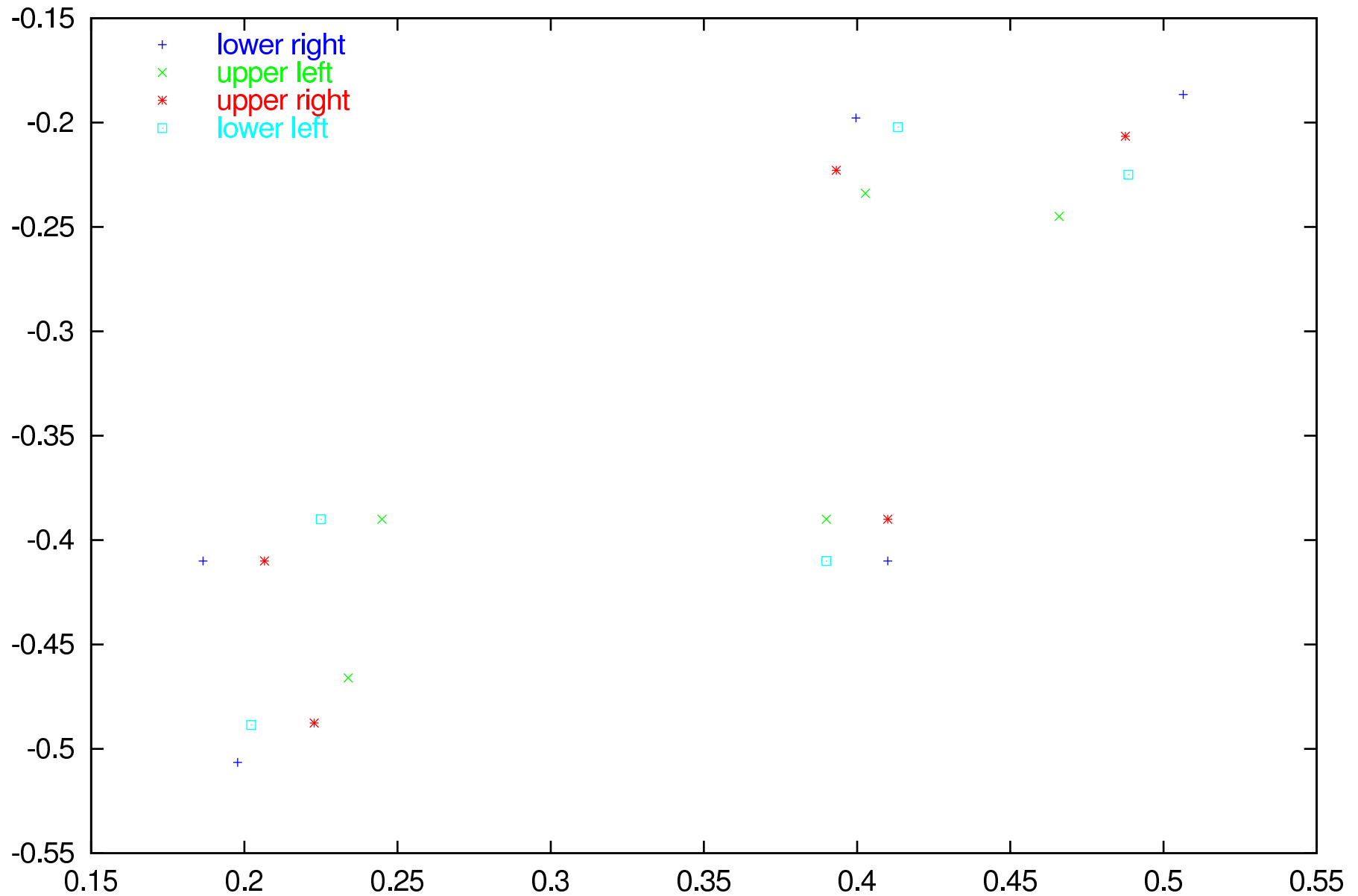
$$\alpha = 2.4 \text{ and } \beta = -1,$$

and concentrate on initial boxes of the form  $(x_0, y_0) \in (0.4, -0.4) + [-d, d]^2$ .

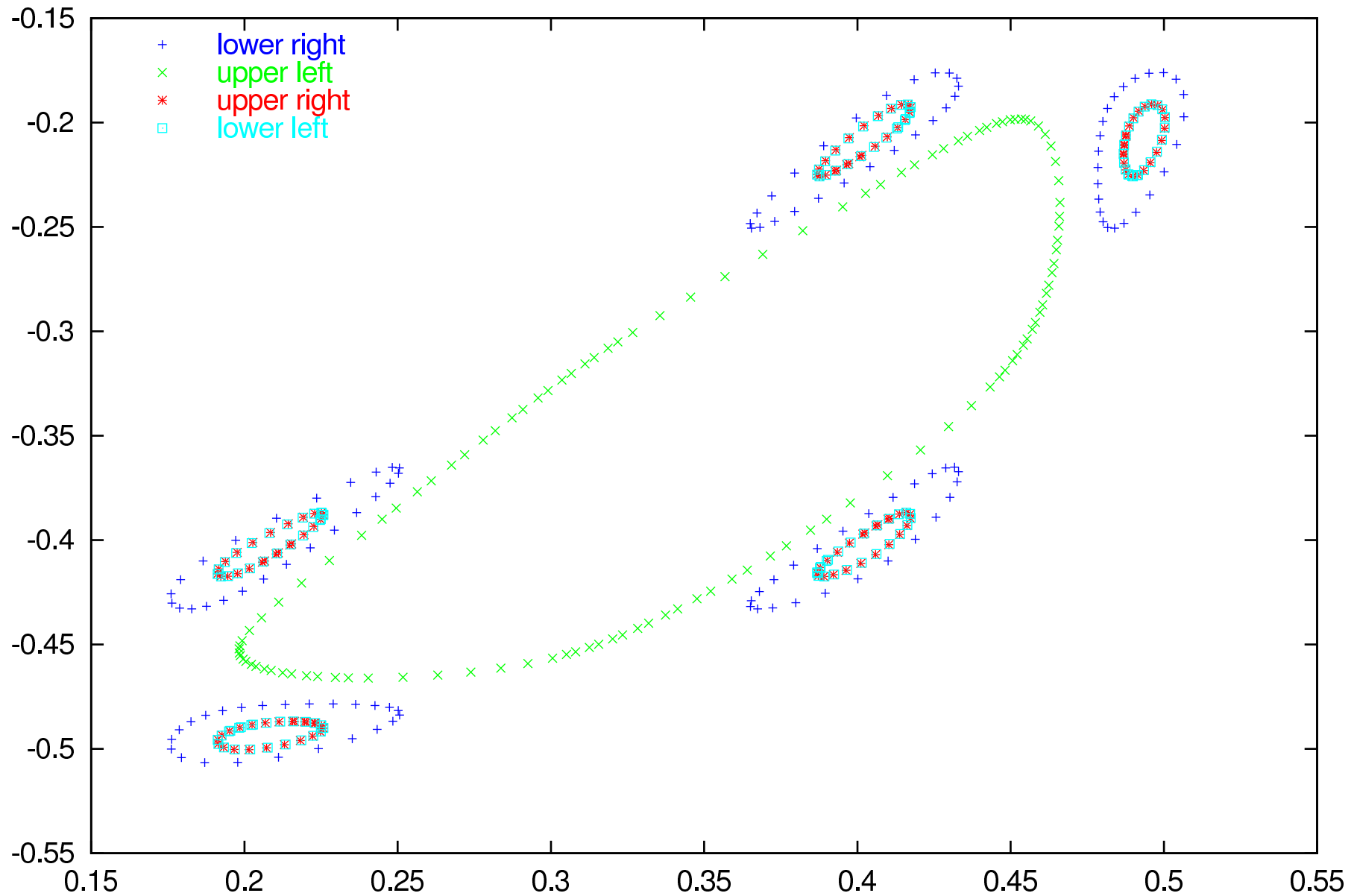
Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , the positions at each step



Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , corner points ( $\pm 0.01$ ) the first 5 steps

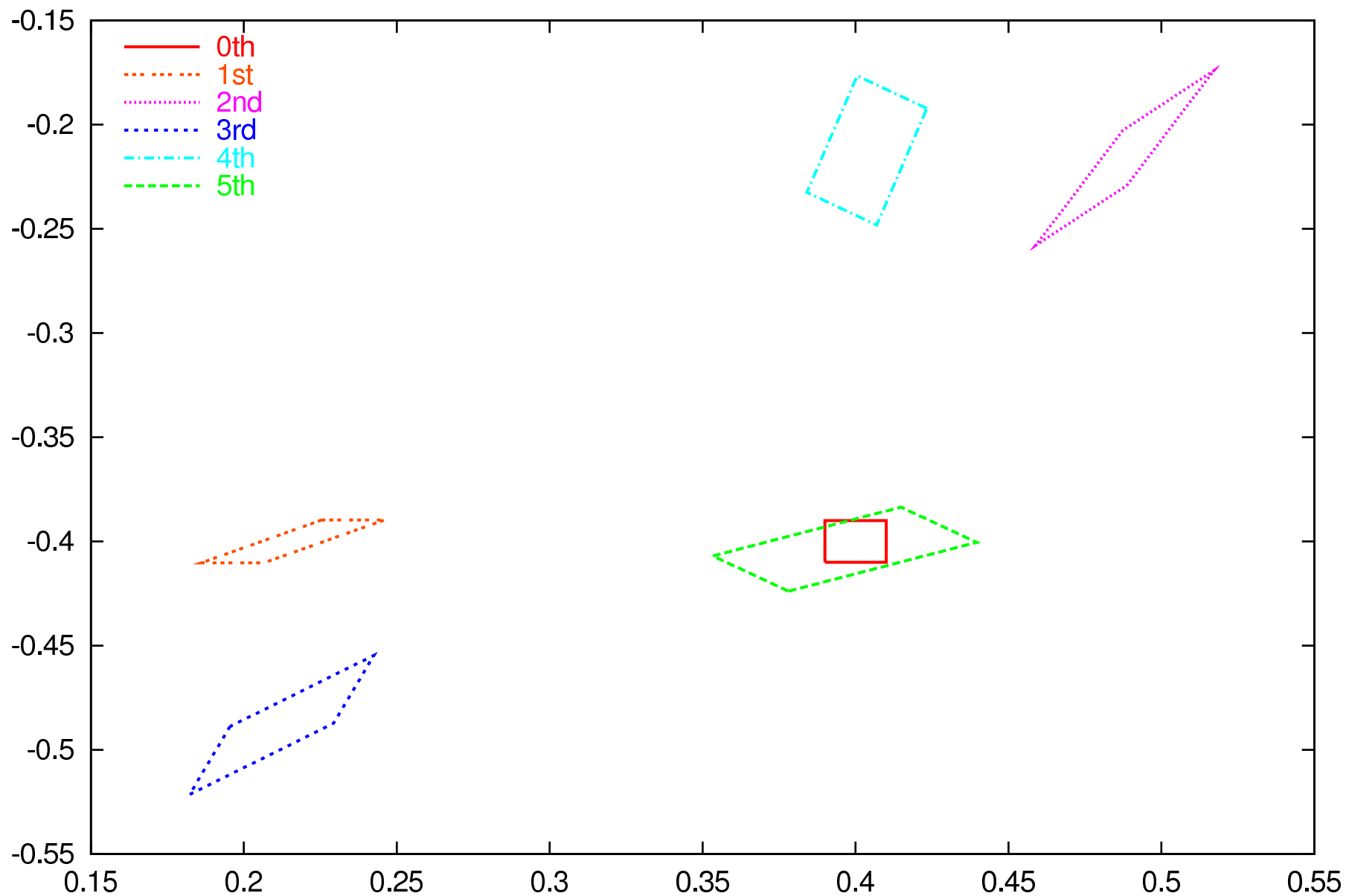


Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , corner points ( $\pm 0.01$ ) the first 120 steps

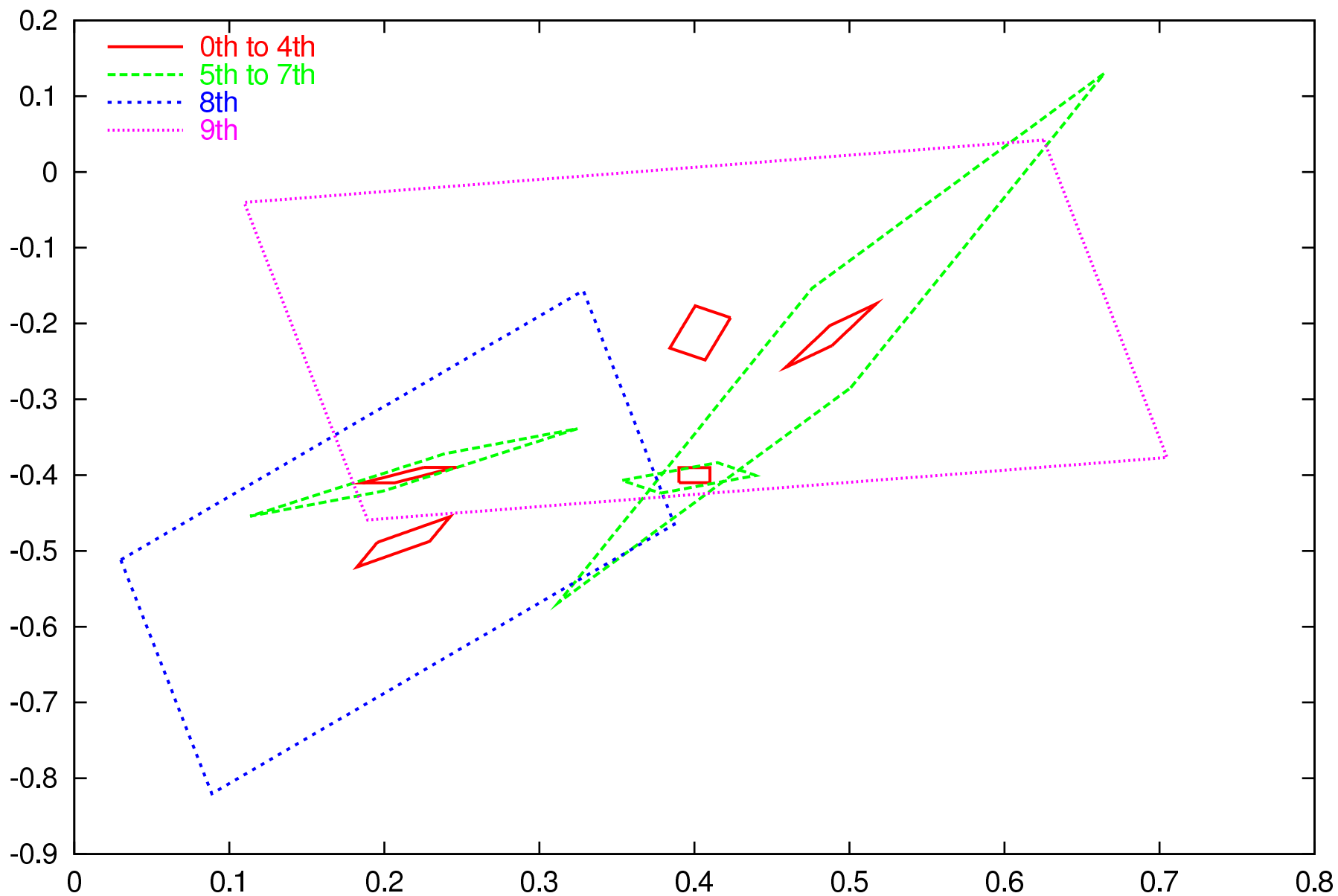




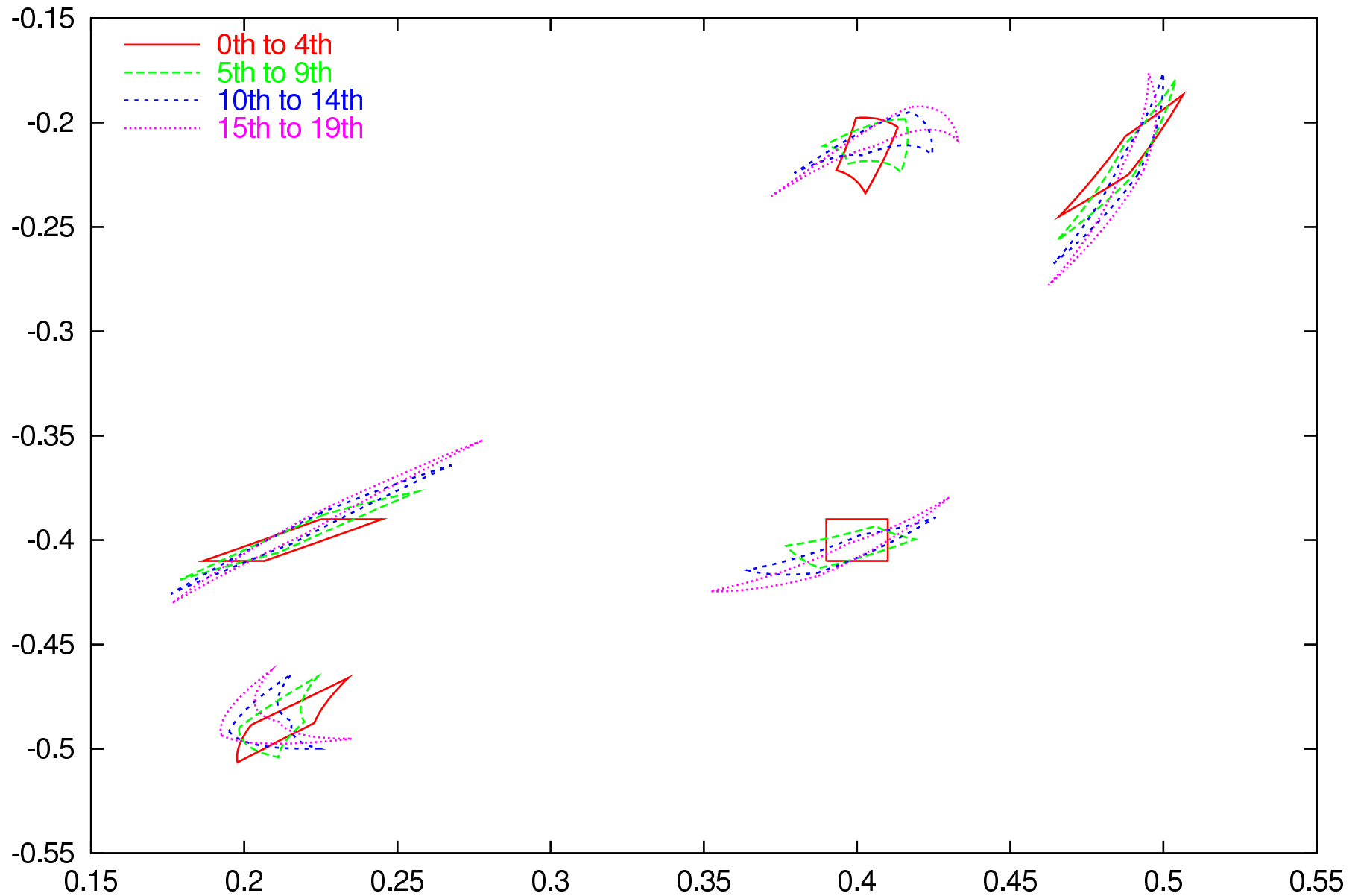
Henon system,  $x_n = 1 - 2.4 \cdot x^2 + y$ ,  $y_n = -x$ , NO=1, SW



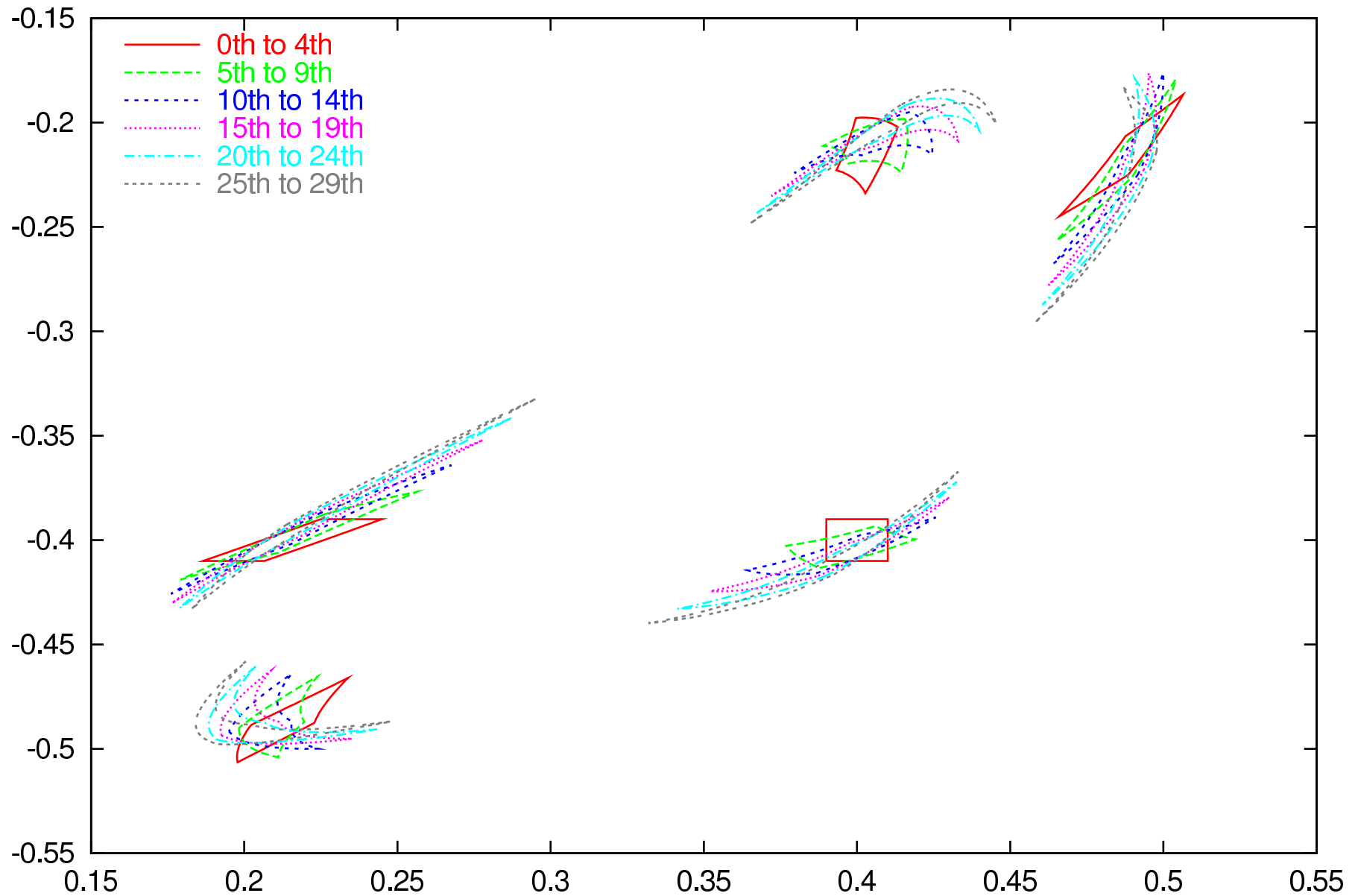
Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=1, SW



Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=20, SW



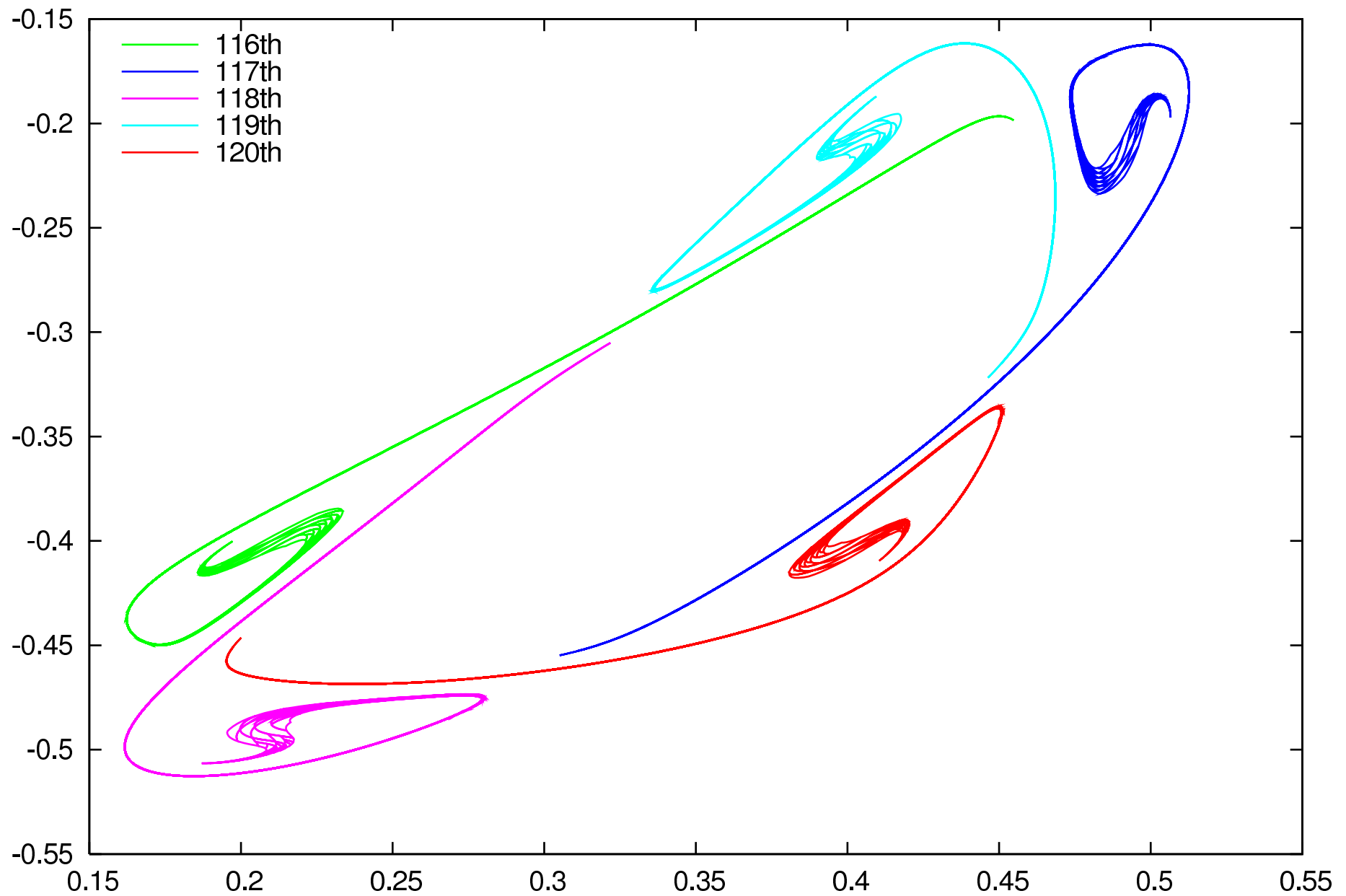
# Henon system, $x_n = 1 - 2.4x^2 + y$ , $y_n = -x$ , NO=20, SW



# Review of the New Features

- The Reference Trajectory and the Flow Operator
- Step Size Control
- Error Parametrization of Taylor Models
- Dynamic Domain Decomposition

Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17



# The Reference Trajectory

**First Step:** Obtain Taylor expansion in time of solution of ODE of center point  $c$ , i.e. obtain

$$c(t) = c_0 + c_1 \cdot (t - t_0) + c_2 \cdot (t - t_0)^2 + \dots + c_n \cdot (t - t_0)^n$$

Very well known from day one how to do this with automatic differentiation. Rather convenient way: can be done by  $n$  iterations of the Picard Operator

$$c(t) = c_0 + \int_0^t f(r(t'), t) dt'$$

in one-dimensional Taylor arithmetic. Each iteration raises the order by one; so in each iteration  $i$ , only need to do Taylor arithmetic in order  $i$ . In either way, this step is **cheap** since it involves only **one-dimensional** operations.

# The Nonlinear Flow

**Second Step:** The goal is to obtain Taylor expansion in time to order  $n$  **and** initial conditions to order  $k$ . Note:

1. This is usually the most **expensive** step. In the original Taylor model-based algorithm, it is done by  $n$  **iterations** of the Picard Operator in multi-dimensional Taylor arithmetic, where  $c_0$  is now a polynomial in initial conditions.
2. The case  $k = 1$  has been known for a long time. Traditionally solved by setting up **ODEs for sensitivities** and solving these as before.
3. The case of higher  $k$  goes back to Beam Physics (M. Berz, Particle Accelerators 1988)
4. Newest Taylor model arithmetic naturally supports different expansions orders  $k$  for initial conditions and  $n$  for time.

**Goal:** Obtain flow with one **single evaluation** of right hand side.



# The Nonlinear Relative ODE

We now develop a better way for second step.

**First:** introduce new "perturbation" variables  $\tilde{r}$  such that

$$r(t) = c(t) + A \cdot \tilde{r}(t).$$

The matrix  $A$  provides **preconditioning**. ODE for  $\tilde{r}(t)$ :

$$\tilde{r}' = A^{-1} [f(c(t) + A \cdot \tilde{r}(t)) - c'(t)]$$

**Second:** evaluate ODE for  $\tilde{r}'$  in Taylor arithmetic. Obtain a Taylor expansion of the ODE, i.e.

$$\tilde{r}' = P(\tilde{r}, t)$$

up to order  $n$  in time and  $k$  in  $\tilde{r}$ . **Very important** for later use: the polynomial  $P$  will have no constant part, i.e.

$$P(0, t) = 0.$$

## Reminder: The Lie Derivative

Let

$$r' = f(r, t)$$

be a dynamical system. Let  $g$  be a variable in state space, and let us study  $g(r(t))$ , i.e. along a solution of the ODE. We have

$$\frac{d}{dt}g(t) = f \cdot \nabla g + \frac{\partial g}{\partial t}$$

Introducing the **Lie Derivative**  $L_f = f \cdot \nabla + \partial/\partial t$ , we have

$$\frac{d^n}{dt^n}g = L_f^n g \text{ and } g(t) \approx \sum_{i=0}^n \frac{(t - t_0)^i}{i!} L_f^i g /_{t=t_0}$$

# Differential Algebras on Taylor Polynomial Spaces

Consider space  ${}_nD_v$  of Taylor polynomials in  $v$  variables and order  $n$  with truncation multiplication. Formally: introduce **equivalence relation** on space of smooth functions

$$f =_n g$$

if all derivatives from 0 to  $n$  agree at 0. **Class** of  $f$  is denoted  $[f]$ . This induces addition, multiplication and scalar multiplication on classes. The resulting structure forms an algebra.

An algebra is a **Differential Algebra** if there is an operation  $\partial$ , called a derivation, that satisfies

$$\begin{aligned}\partial(s \cdot a + t \cdot b) &= s \cdot \partial a + t \cdot \partial b \text{ and} \\ \partial(a \cdot b) &= a \cdot (\partial b) + (\partial a) \cdot b\end{aligned}$$

for any vectors  $a$  and  $b$  and scalars  $s$  and  $t$ . **Unfortunately**, the **natural partial derivative** operations  $[f] \rightarrow [\partial_i f]$  does **not** introduce a differential algebra, because of loss of highest order.

# Differential Algebras on Taylor Polynomial Spaces

However, consider the modified operation

$$\partial_f \text{ with } \partial_f g = f \cdot \nabla g$$

If  $f$  is origin preserving, i.e.  $f(0) = 0$ , then  $\partial_f$  is a derivation on the space  ${}_n D_v$ . Why?

- Each derivative operation in the gradient  $\nabla g$  loses the highest order;
- but since  $f(0) = 0$ , the missing order in  $\nabla g$  **does not matter** since it does not contribute to the product  $f \cdot \nabla g$ .

# Polynomial Flow from Lie Derivative

Remember the ODE for  $\tilde{r}'$ :

$$\tilde{r}' = P(\tilde{r}, t)$$

up to order  $n$  in time and  $k$  in  $\tilde{r}$ . And remember  $P(0, t) = 0$ . Thus we can obtain the  $n$ -th order expansion of the flow as

$$\tilde{r}(t) = \sum_{i=0}^n \frac{(t - t_0)^i}{i!} \cdot \left( P \cdot \nabla + \frac{\partial}{\partial t} \right)^i \tilde{r}_0 \Bigg|_{t=t_0}$$

- The fact that  $P(0, t) = 0$  restores the derivatives lost in  $\nabla$
- The fact that  $\partial/\partial t$  appears without origin-preserving factor limits the expansion to order  $n$ .

# Performance of Lie Derivative Flow Methods

Apparently we have the following:

- Each term in the Lie derivative sum requires  $v + 1$  derivations (very cheap, just re-shuffling of coefficients)
- Each term requires  $v$  multiplications
- We need **one** evaluation of  $f$  in  ${}_nD_v$  (to set up ODE)

Compare this with the conventional algorithm, which requires  $n$  evaluations of the function  $f$  of the right hand side. Thus, roughly, if the evaluation of  $f$  requires more than  $v$  multiplications, the new method is more efficient.

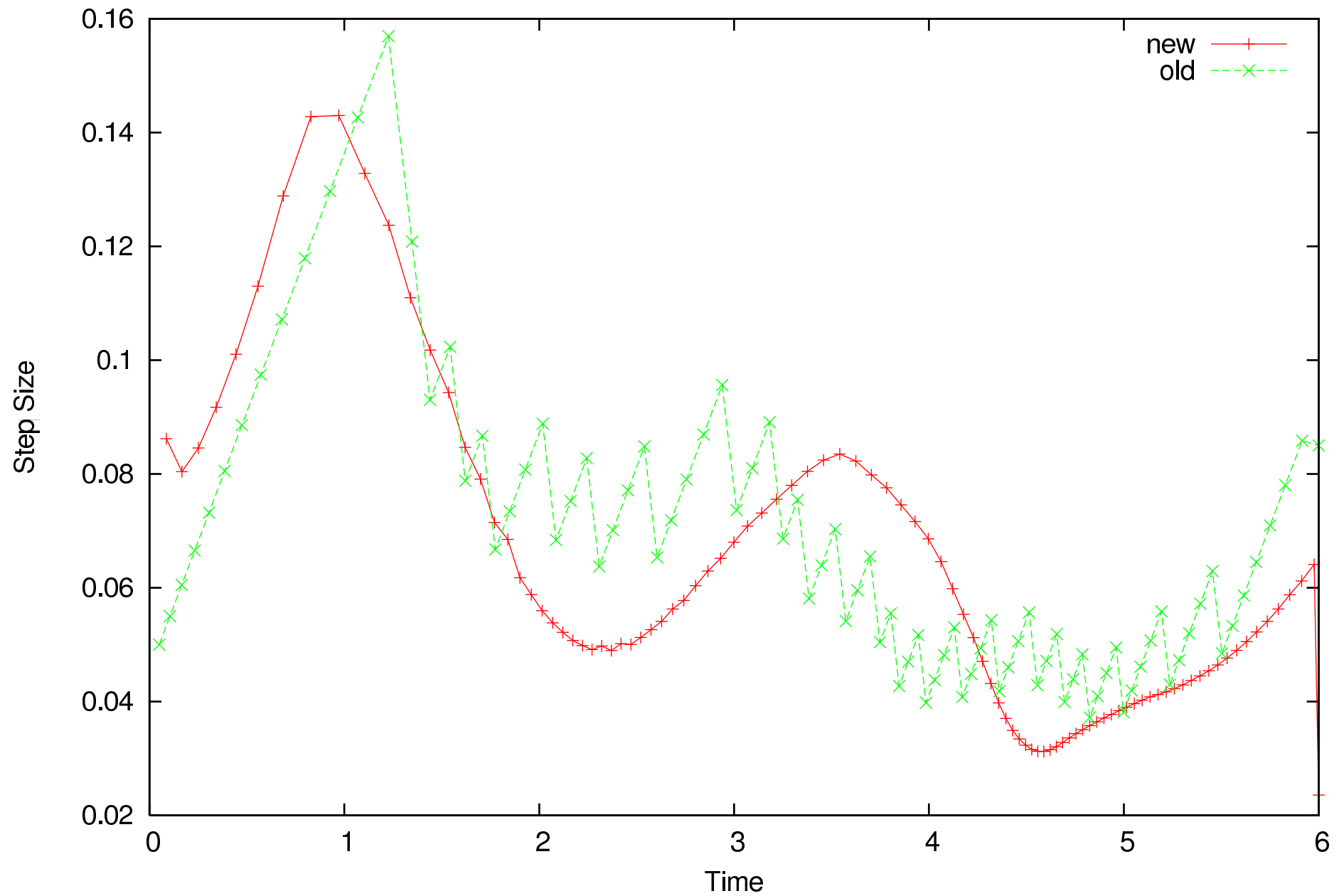
- Many practically appearing right hand sides  $f$  satisfy this.
- But on the other hand, if the function  $f$  does not satisfy this (for example for the linear case), then also  $P$  will be simple (in the linear case:  $P$  will be linear), and thus less operations appear

# Step Size Control

Step size control to maintain approximate error  $\varepsilon$  in each step.  
Based on a suite of tests:

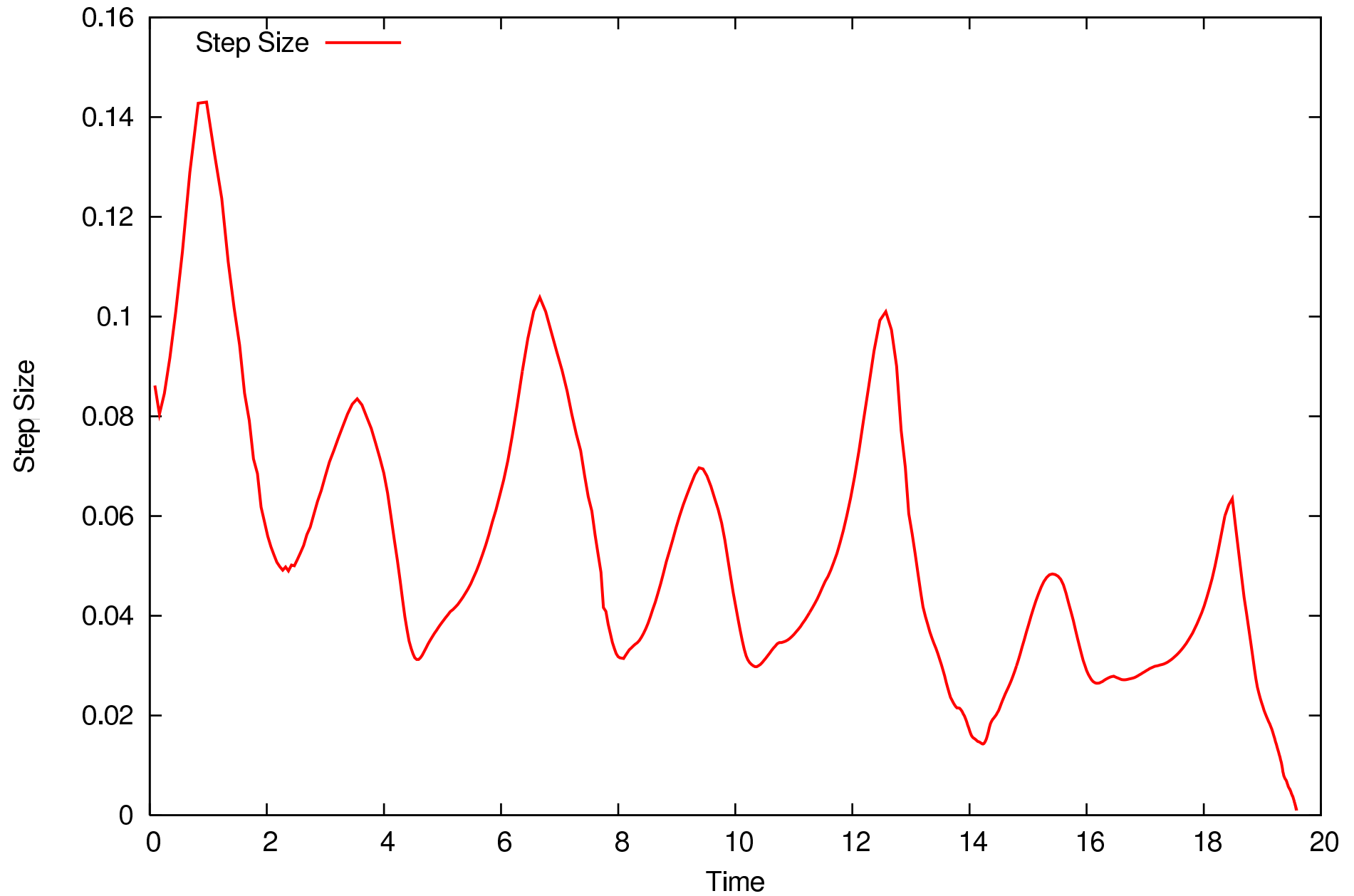
1. Utilize the **Reference Orbit**. Extrapolate the size of coefficients for estimate of remainder error, scale so that it reaches and get  $\Delta t_1$ . Goes back to Moore in 1960s. This is one of conveniences when using Taylor integrators.
2. Utilize the **Flow**. Compute flow time step with  $\Delta t_1$ . Extrapolate the contributions of each order of flow for estimate of remainder error to get update  $\Delta t_2$ .
3. Utilize a **Correction factor**  $c$  to account for overestimation in TM arithmetic as  $c = \sqrt[n+1]{|R|/\varepsilon}$ . Largely a measure of complexity of ODE. Dynamically update the correction factor.
4. Perform verification attempt for  $\Delta t_3 = c \cdot \Delta t_2$

Roessler NO=18, (new code: eps=1e-13, old code: TOL=1e-9)





COSY-VI Roessler until Break-down, Step Size, April 13 2007



# Error Parametrization of Taylor models

**Motivation:** Is it possible to absorb the remainder error bound intervals of Taylor models into the polynomial parts using additional parameters?

Phrase the question as the following problem:

1. Have Taylor models with 0 remainder error interval, which depend on the independent variables  $\vec{x}$  and the parameters  $\vec{\alpha}$ .

$$\vec{T}_0 = \vec{P}_0(\vec{x}, \vec{\alpha}) + \overrightarrow{[0, 0]}.$$

2. Perform Taylor model arithmetic on  $\vec{T}_0$ , namely  $\vec{F}(\vec{T}_0)$

$$\vec{F}(\vec{T}_0) = \vec{P}(\vec{x}, \vec{\alpha}) + \vec{I}_F, \text{ where } \vec{I}_F \neq \overrightarrow{[0, 0]}.$$

3. Try to absorb  $\vec{I}_F$  into the polynomial part that depends on  $\vec{\alpha}$

$$\vec{P}(\vec{x}, \vec{\alpha}) + \vec{I}_F \subseteq \vec{P}'(\vec{x}, \vec{\alpha}) + \overrightarrow{[0, 0]}. \quad (\text{A})$$

## Observe

$$\vec{P}(\vec{x}, \vec{\alpha}) = \underbrace{\vec{P}(\vec{x}, 0)}_{\vec{\alpha}\text{-indep.}} + \underbrace{\vec{P}(\vec{x}, \vec{\alpha}) - \vec{P}(\vec{x}, 0)}_{\vec{\alpha}\text{-dependent}} = \vec{P}(\vec{x}, 0) + \vec{P}_\alpha(\vec{x}, \vec{\alpha})$$

The size of  $\vec{P}(\vec{x}, 0)$  is much larger than the rest, because the rest is essentially errors. The process of (A) does not alter  $\vec{P}(\vec{x}, 0)$ , so set the  $\vec{\alpha}$ -independent part  $\vec{P}(\vec{x}, 0)$  aside from the whole process, which helps the numerical stability of the process.

The task is now

$$\vec{P}_\alpha(\vec{x}, \vec{\alpha}) + \vec{I}_F \subseteq \vec{P}'_\alpha(\vec{x}, \vec{\alpha}) + \overrightarrow{[0, 0]}.$$

We limit  $\vec{P}_\alpha(\vec{x}, \vec{\alpha})$  to be only **linearly** dependent on  $\vec{\alpha}$ .

$$\vec{P}_\alpha(\vec{x}, \vec{\alpha}) + \vec{I}_F = \left( \widehat{M} + \widehat{M}(\vec{x}) \right) \cdot \vec{\alpha} + \vec{I}_F.$$

Express  $\vec{I}_F$  by the matrix form using additional parameters  $\vec{\beta}$

$$\vec{I}_F \subseteq \left( \widehat{I}_F + \widehat{\bar{I}}_F(\vec{x}) \right) \cdot \vec{\beta}.$$

where  $\widehat{\bar{I}}_F(\vec{x}) = 0$  and  $\left( \widehat{I}_F \right)_{ii} = |I_{Fi}|$ .

$$\vec{P}_\alpha(\vec{x}, \vec{\alpha}) + \vec{I}_F \subseteq \left( \widehat{M} + \widehat{\bar{M}}(\vec{x}) \right) \cdot \vec{\alpha} + \left( \widehat{I}_F + \widehat{\bar{I}}_F(\vec{x}) \right) \cdot \vec{\beta}.$$

View this as a collection of  $2 \cdot v$  column vectors associated to  $2 \cdot v$  parameters  $\vec{\alpha}$  and  $\vec{\beta}$ . Recall a matrix, or a collection of  $v$  column vectors, represent a parallelepiped. The problem is now to find a **set sum of two parallelepipeds**.

# Psum Algorithm for choosing column vectors

**Task:** Choose  $v$  vectors out of  $n$  vectors  $\vec{s}_i$ ,  $i = 1, \dots, n$ ,  $n \geq v$ .

1. Choose the longest vector  $\vec{s}_k$ , and assign it as  $\vec{t}_1$ . Normalize it as  $\vec{e}_1 = \vec{t}_1 / |\vec{t}_1|$ .
2. Out of the remaining vectors  $\vec{s}_i$ , choose the  $j$ -th vector  $\vec{t}_j = \vec{s}_k$  such that

$$\frac{|\vec{s}_k|^2 - \sum_{m=1}^{j-1} |\vec{s}_k \cdot \vec{e}_m|^2}{|\vec{s}_k|^{2p}}$$

is largest. Compute  $\vec{e}_j$ , the orthonormalized vector of  $\vec{t}_j$  to  $\vec{e}_1, \dots, \vec{e}_{j-1}$ . (Gram-Schmidt)

3. Repeat the process 2 until  $j = v$ .

Experimentally,  $p = 0.5$  is found to be efficient and robust for obtaining a set sum of two parallelepipeds

## Psum Algorithm for two parallelepipeds

**Task:** Obtain a set sum of two parallelepipeds  $\widehat{M}_1$  and  $\widehat{M}_2$ .

1. Prepare the basis  $\widehat{M}_b$  using the Psum algorithm for choosing  $v$  column vectors out of  $2 \cdot v$  column vectors from  $\widehat{M}_1$  and  $\widehat{M}_2$ .
2. Compute conditioned parallelepipeds  $\widehat{M}_b^{-1} \cdot \widehat{M}_1$  and  $\widehat{M}_b^{-1} \cdot \widehat{M}_2$ .
3. Confine the conditioned parallelepipeds by bounding them.

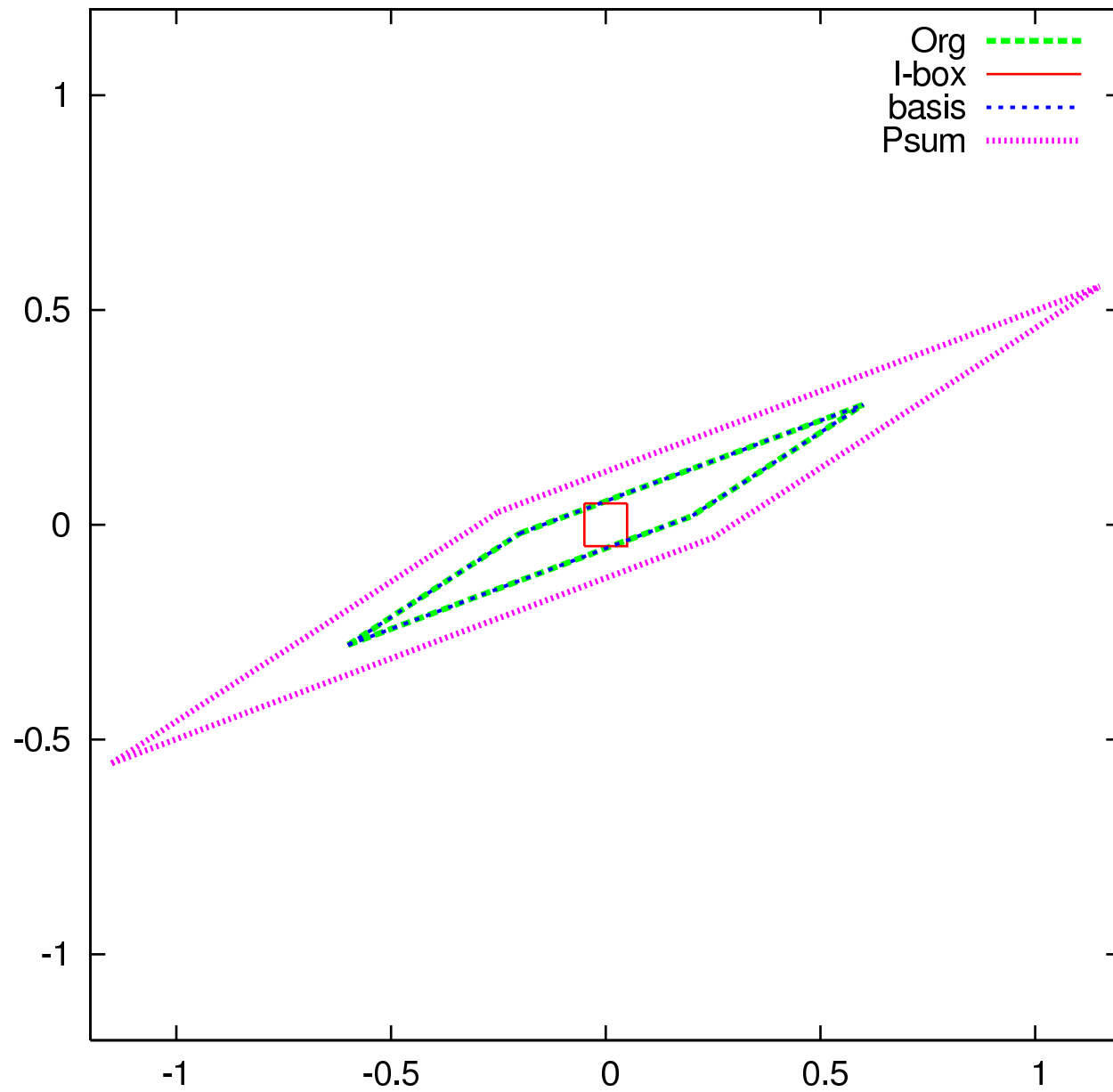
$$\vec{B}_1 = \text{bound} \left( \widehat{M}_b^{-1} \cdot \widehat{M}_1 \right) \text{ and } \vec{B}_2 = \text{bound} \left( \widehat{M}_b^{-1} \cdot \widehat{M}_2 \right).$$

4. Compute the interval sum  $\vec{B} = \vec{B}_1 + \vec{B}_2$ .  $\vec{B}$  confines the conditioned set sum of the conditioned parallelepipeds.

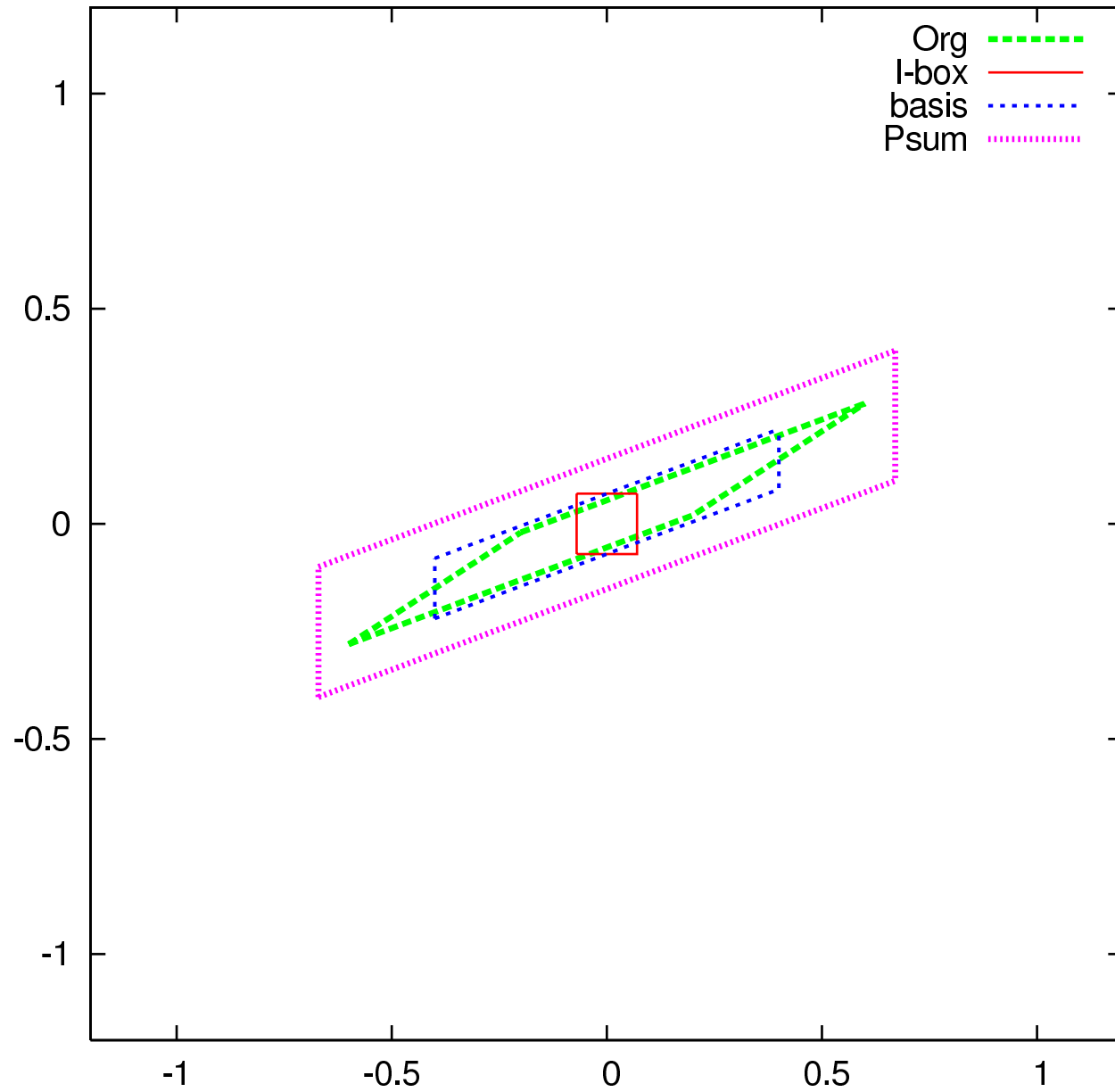
5. From  $\vec{B}$ , set up a parallelepiped as a box  $\widehat{B} = \begin{pmatrix} |B_1| & \dots & 0 \\ 0 & & |B_v| \end{pmatrix}.$

6. Compute  $\widehat{M}_b \cdot \widehat{B}$ , which is a set sum of  $\widehat{M}_1$  and  $\widehat{M}_2$  under  $\widehat{M}_b$ .

Psum of Org Parallelpiped (0.4,0.15)-(0.2,0.13) and I-box 0.05-0.05



Psum of Org Parallelpiped (0.4,0.15)-(0.2,0.13) and I-box 0.07-0.07





## Error Absorption

We now chose a favoured collection of  $v$  column vectors  $\widehat{L} + \widehat{\widehat{L}}(\vec{x})$  using the Psum algorithm. Collect the left over  $v$  column vectors to  $\widehat{E} + \widehat{\widehat{E}}(\vec{x})$ . Associate them to  $2 \cdot v$  parameters  $\vec{\alpha}'$  and  $\vec{\beta}'$ .

$$\vec{P}_\alpha(\vec{x}, \vec{\alpha}) + \vec{I}_F \subseteq \left( \widehat{L} + \widehat{\widehat{L}}(\vec{x}) \right) \cdot \vec{\alpha}' + \left( \widehat{E} + \widehat{\widehat{E}}(\vec{x}) \right) \cdot \vec{\beta}'.$$

Since  $\vec{\alpha}$  and  $\vec{\beta}$  do not appear anymore, we can rename  $\vec{\alpha}'$  and  $\vec{\beta}'$  as  $\vec{\alpha}$  and  $\vec{\beta}$  for the simplicity.

$$\begin{aligned} \vec{P}_\alpha(\vec{x}, \vec{\alpha}) + \vec{I}_F &\subseteq \left( \widehat{L} + \widehat{\widehat{L}}(\vec{x}) \right) \cdot \vec{\alpha} + \left( \widehat{E} + \widehat{\widehat{E}}(\vec{x}) \right) \cdot \vec{\beta} \\ &= \widehat{L} \circ \left[ \widehat{L}^{-1} \circ \left( \widehat{L} + \widehat{\widehat{L}}(\vec{x}) \right) \cdot \vec{\alpha} + \widehat{L}^{-1} \circ \left( \widehat{E} + \widehat{\widehat{E}}(\vec{x}) \right) \cdot \vec{\beta} \right] \\ &\subseteq \widehat{L} \circ \left[ \left( \widehat{I} + \widehat{L}^{-1} \circ \widehat{\widehat{L}}(\vec{x}) \right) \cdot \vec{\alpha} + \widehat{B} \cdot \vec{\beta} \right] \end{aligned}$$

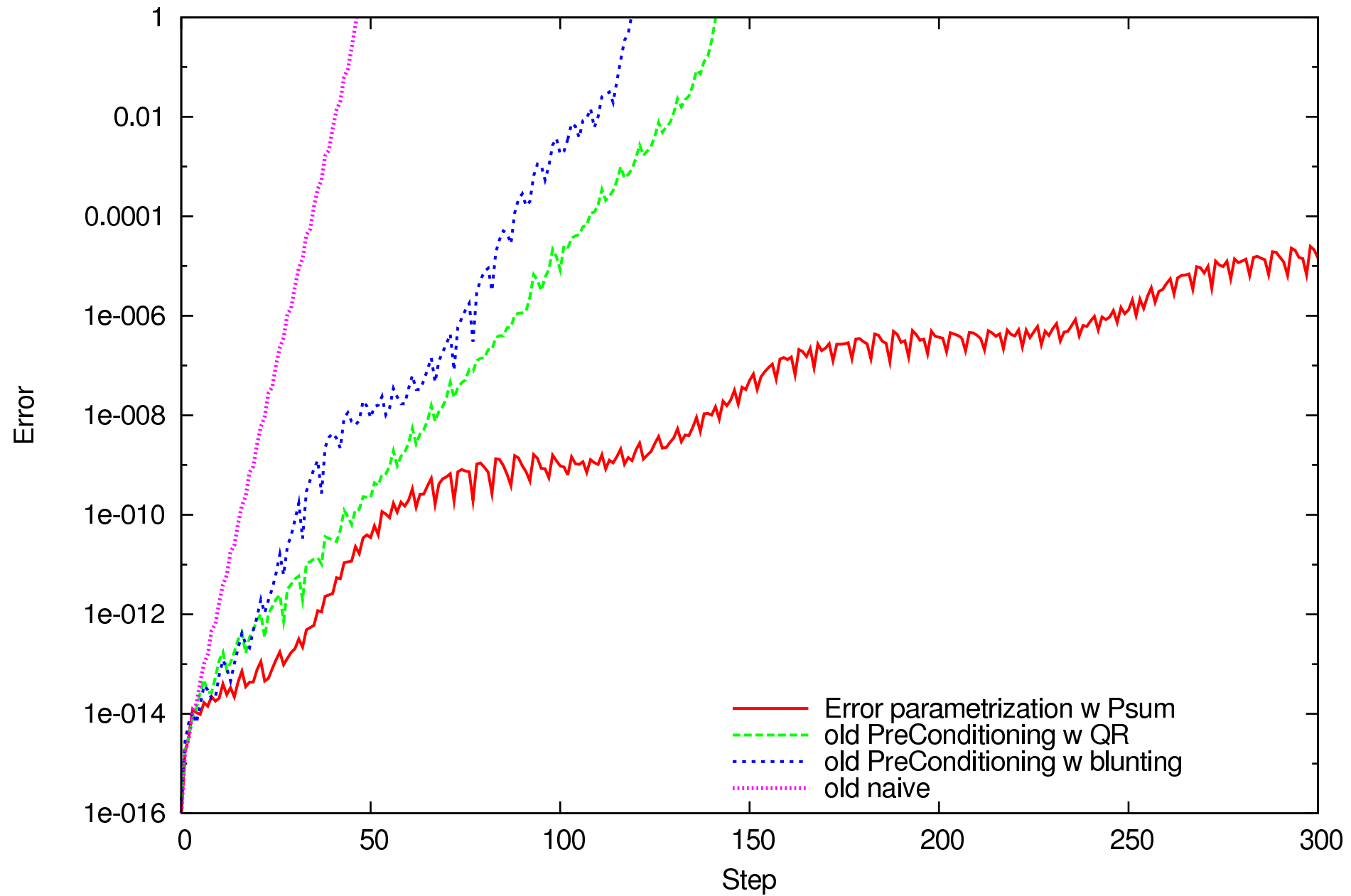
where  $\widehat{B}$  is a diagonal matrix with the  $i$ -th element is  $|B_i|$  and  $\vec{B} = \text{bound} \left( \widehat{L}^{-1} \circ \left( \widehat{E} + \widehat{\widehat{E}}(\vec{x}) \right) \cdot \vec{\beta} \right)$ .

**If the diagonal terms of  $\left(\widehat{I} + \widehat{L}^{-1} \circ \widehat{\widehat{L}}(\vec{x})\right)$  are positive,**

$$\begin{aligned}
\vec{P}_\alpha(\vec{x}, \vec{\alpha}) + \vec{I}_F &\subseteq \widehat{L} \circ \left[ \left( \widehat{I} + \widehat{L}^{-1} \circ \widehat{\widehat{L}}(\vec{x}) \right) \cdot \vec{\alpha} + \widehat{B} \cdot \vec{\alpha} \right] \\
&= \widehat{L} \circ \left( \widehat{I} + \widehat{L}^{-1} \circ \widehat{\widehat{L}}(\vec{x}) \right) \cdot \vec{\alpha} + \widehat{L} \circ \widehat{B} \cdot \vec{\alpha} \\
&= \left( \widehat{L} + \widehat{\widehat{L}}(\vec{x}) + \widehat{L} \circ \widehat{B} \right) \cdot \vec{\alpha}.
\end{aligned}$$

**Note:** A modification to use  $\widehat{A}$  instead of  $\widehat{L}$ , when  $\widehat{A} \approx \widehat{L}$ , is done easily. This involves bounding of  $\widehat{A}^{-1} \circ \left( \widehat{L} - \widehat{A} \right) \cdot \vec{\alpha}$  and the diagonal terms to be checked positive are those of  $\left( \widehat{I} + \widehat{A}^{-1} \circ \widehat{\widehat{L}}(\vec{x}) \right)$ .

henon (area preserving). Performance Comparison. TM order 13, IC width 4e-3



# Cost of Additional Parameters

For a  $v$  dimensional system, we need  $v$  parameters  $\vec{\alpha}$  to absorb Taylor model remainder error bound intervals. The dependence on  $\vec{\alpha}$  is limited to **linear**. So, we use weighted DA. Choose an appropriate weight order  $w$  for  $\vec{\alpha}$ .

- The dependence on  $\vec{\alpha}$  has to be kept linear. Namely  $2 \cdot w > n$ , where  $n$  is the computational order of Taylor models. Choose

$$w = \text{Int} \left( \frac{n}{2} \right) + 1.$$

Maximum size necessary for DA and TM for  $v = 2$ .

$n$	$v$	DA	TM			$v$	DA	TM			$w$	$v_w$	DA	TM
13	2	105	140			$2 + 2$	2380	2419			7	$2 + 2_w$	161	200
21	2	253	304			$2 + 2$	12650	12705	$\Rightarrow$		11	$2 + 2_w$	385	440
33	2	595	670			$2 + 2$	66045	66124			17	$2 + 2_w$	901	980

# Dynamic Domain Decomposition

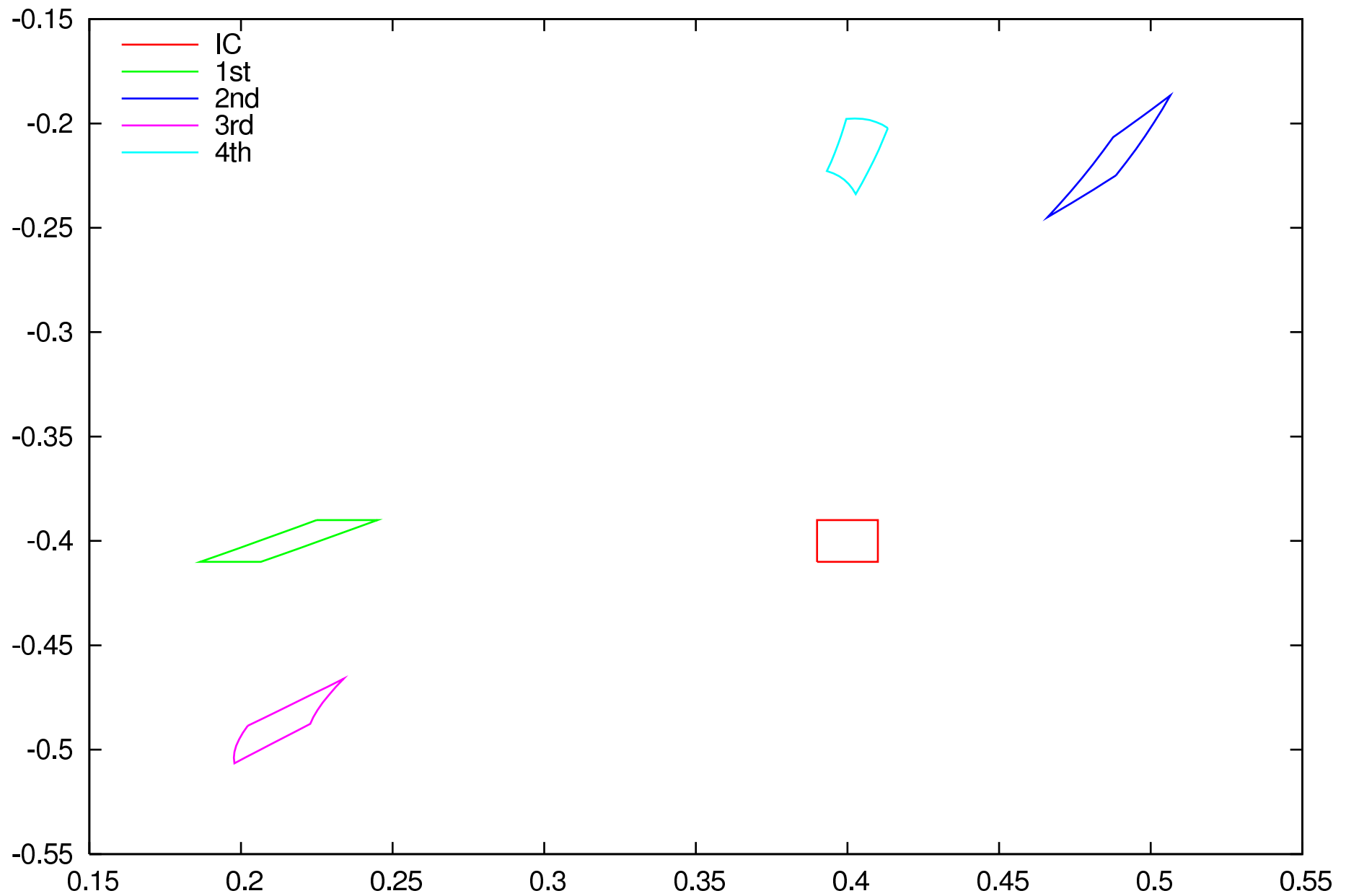
For extended domains, this is **natural equivalent** to step size control. Similarity to what's done in global optimization.

1. Evaluate ODE for  $\Delta t = 0$  for current flow.
2. If resulting remainder bound  $R$  greater than  $\varepsilon$ , split the domain along variable leading to longest axis.
3. Absorb  $R$  in the TM polynomial part using the error parametrization method. If it fails, split the domain along variable leading to largest  $x$  dependence of the error.
4. Put one half of the box on stack for future work.

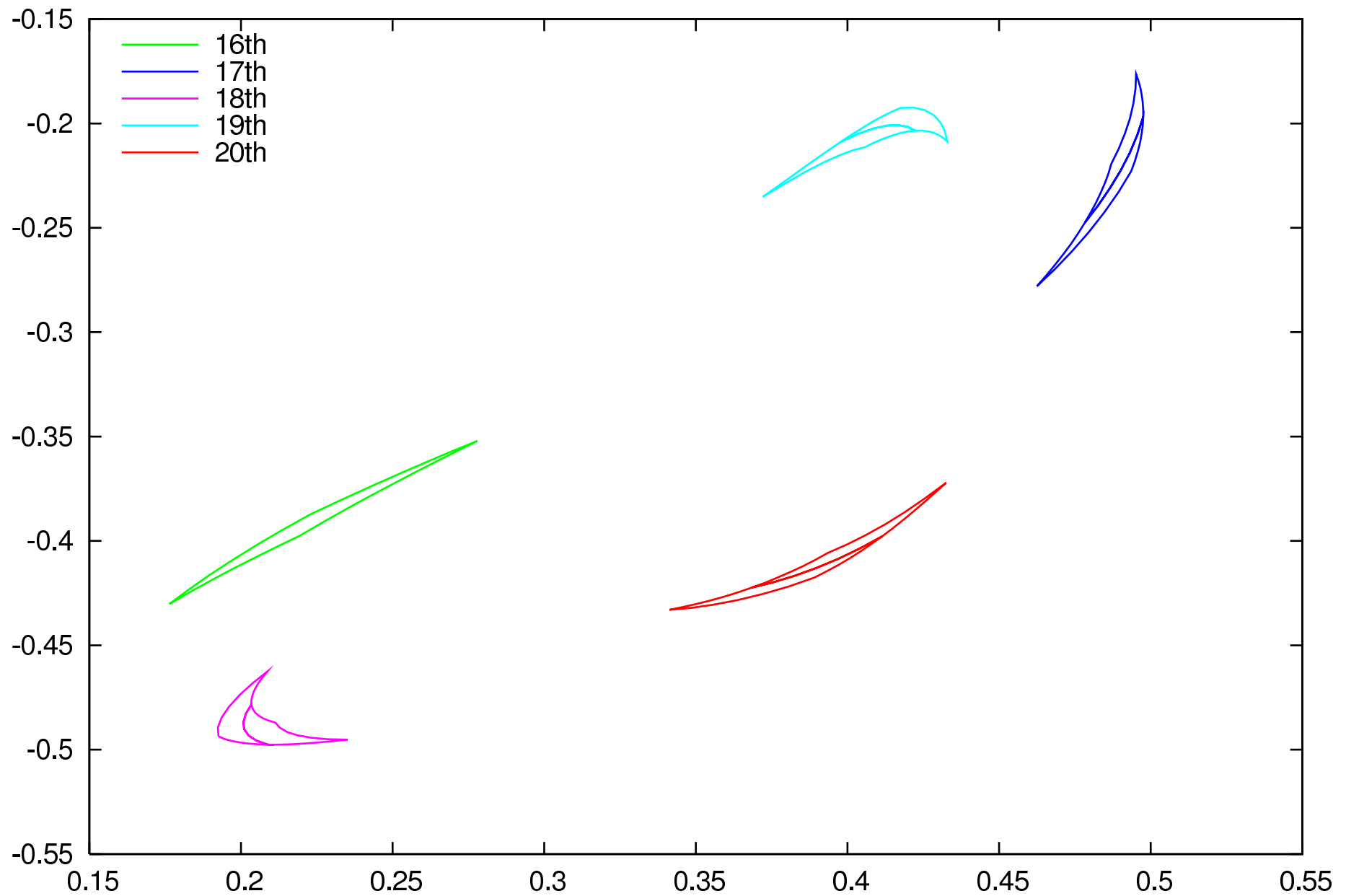
Things to consider:

- Utilize "First-in-last-out" stack; minimizes stack length. Special adjustments for stack management in a parallel environment, including load balancing.
- Outlook: also dynamic order control for dependence on initial conditions

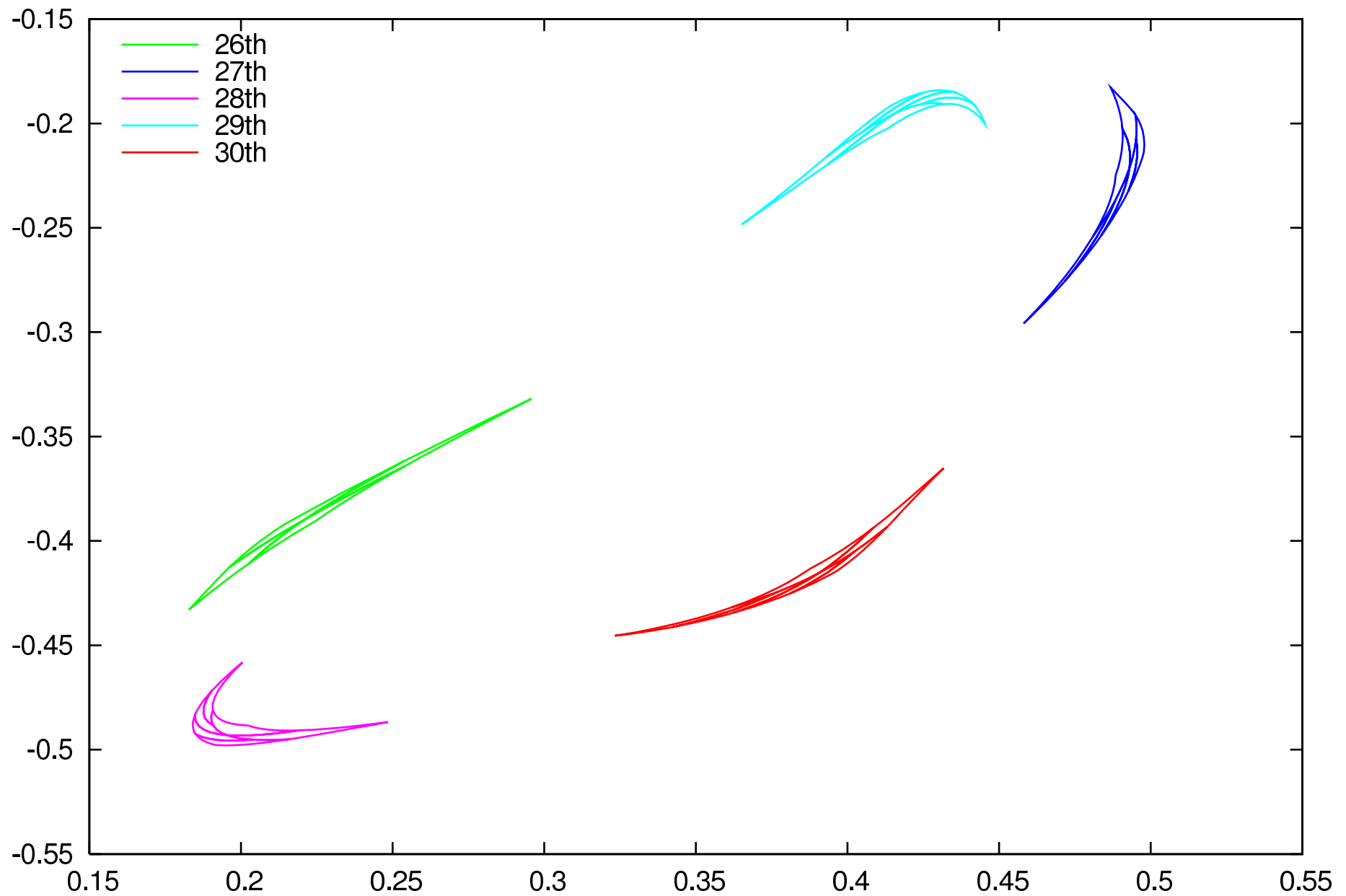
Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17



Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17

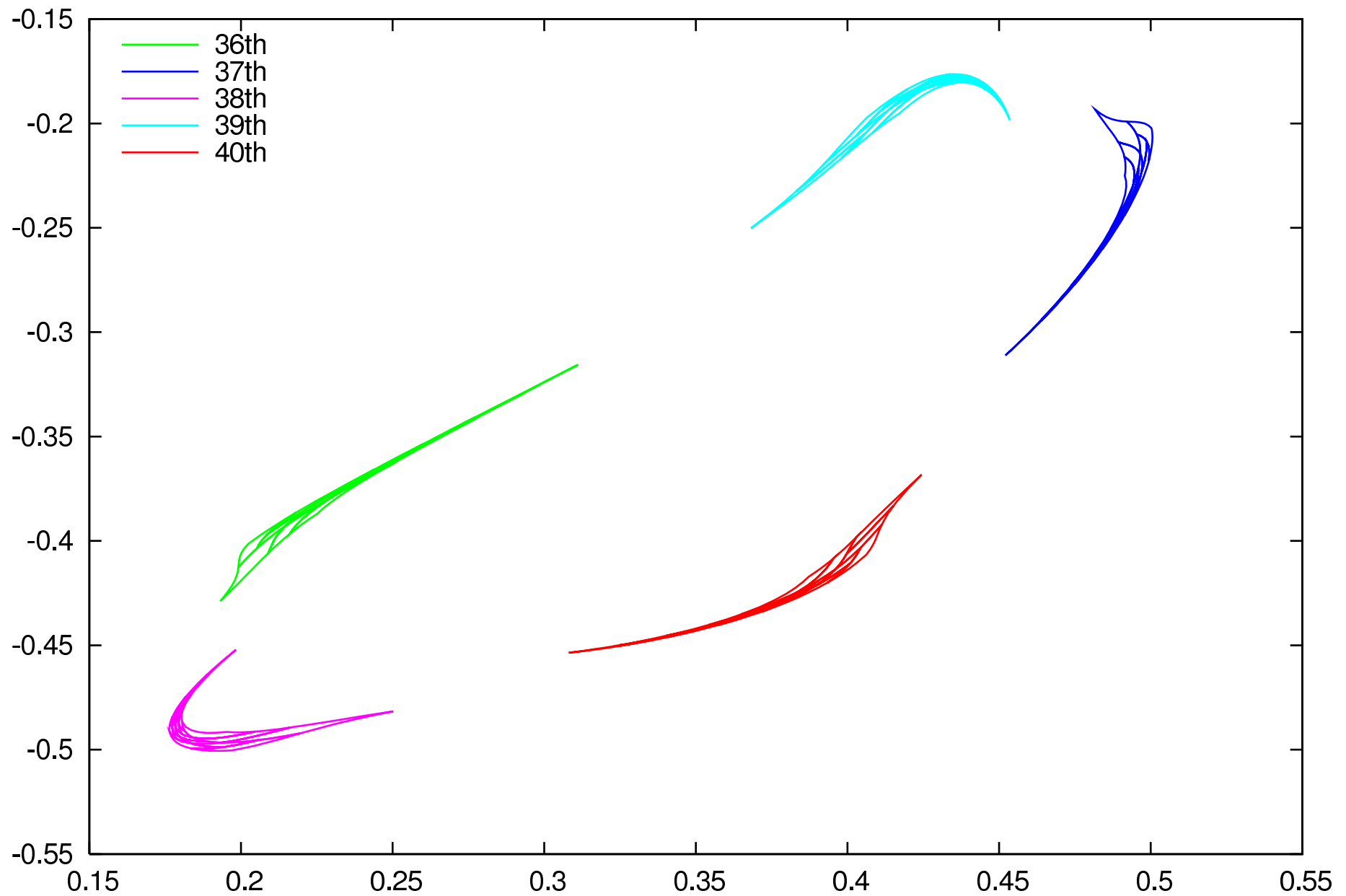


Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17

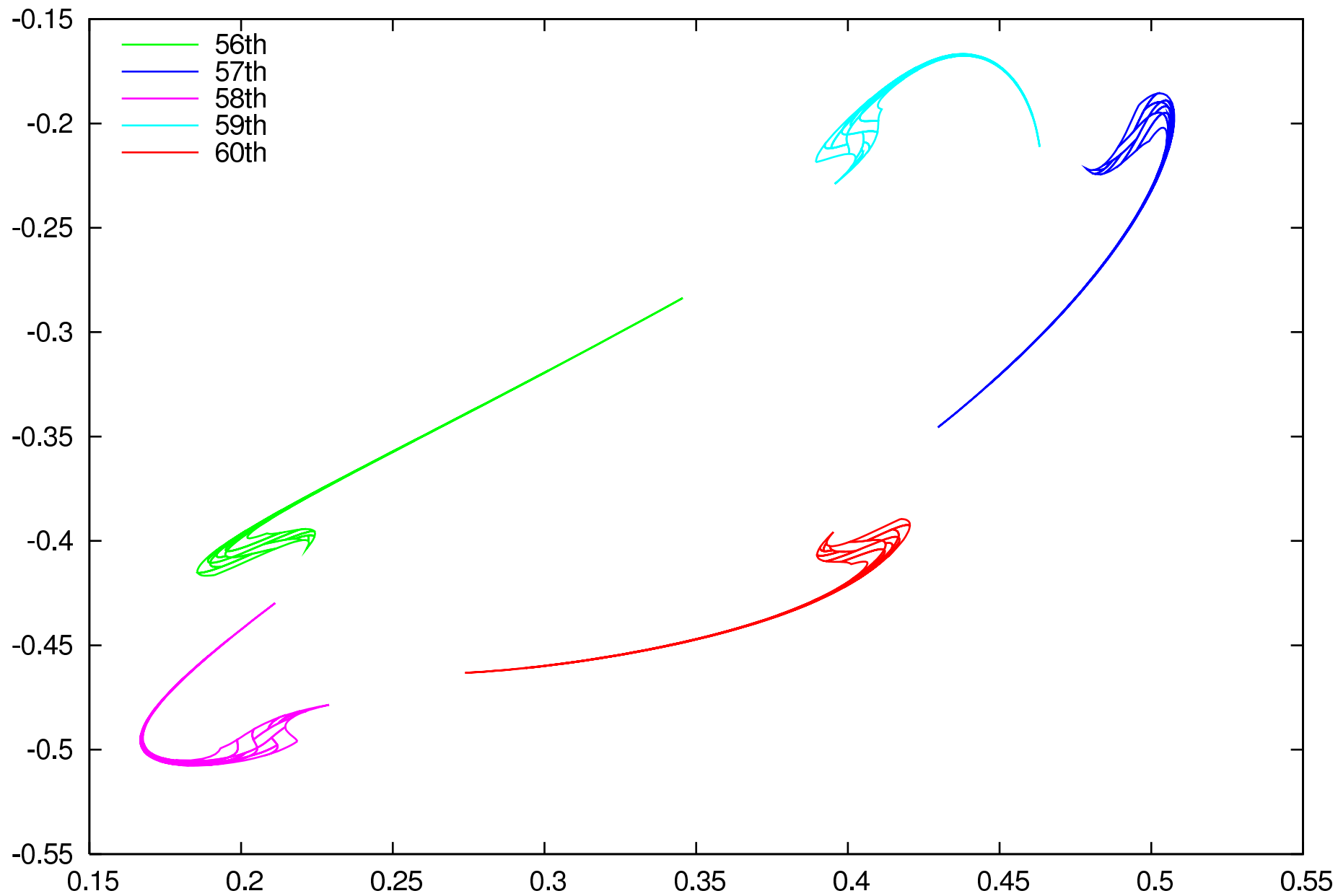




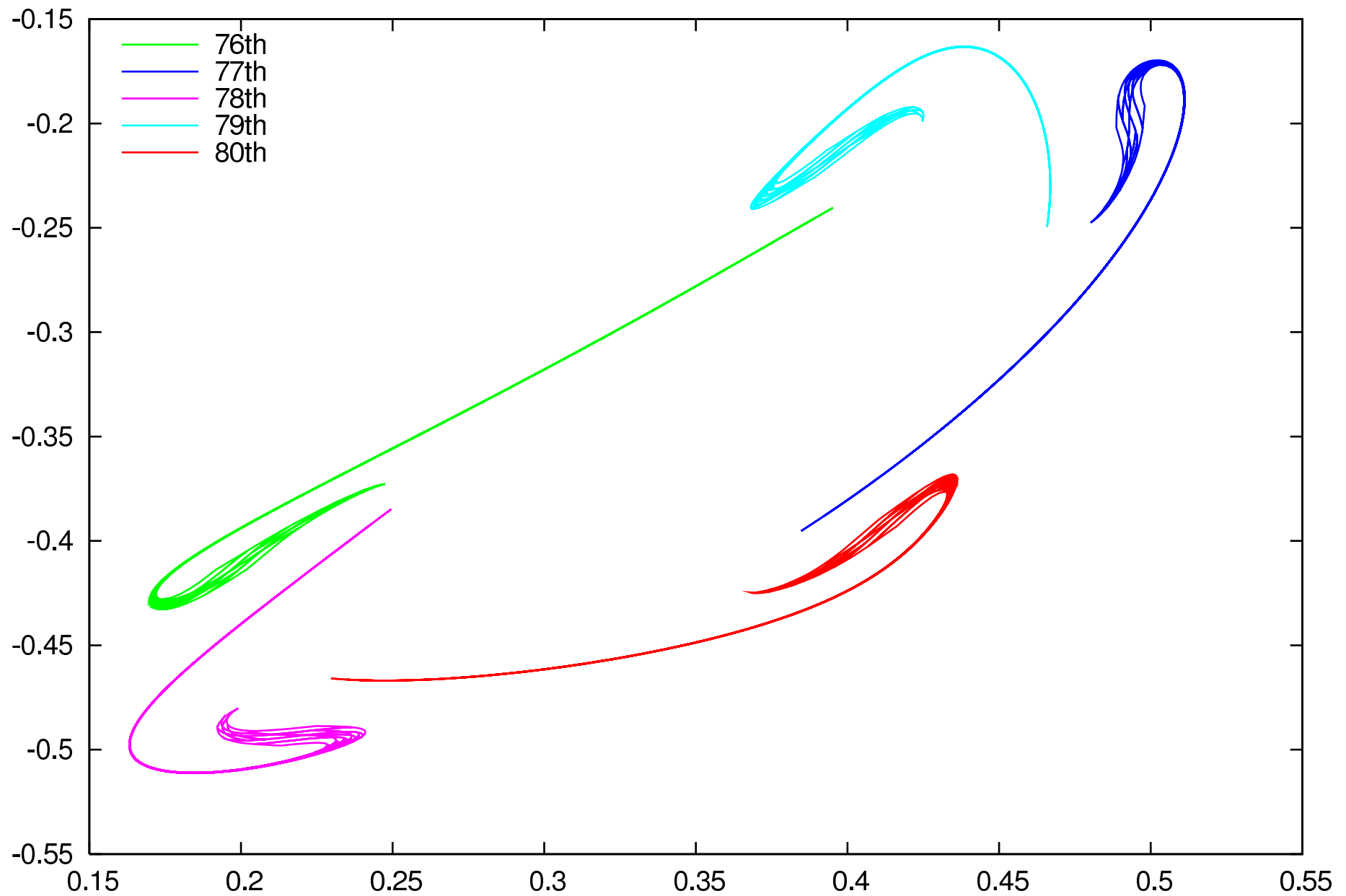
Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17



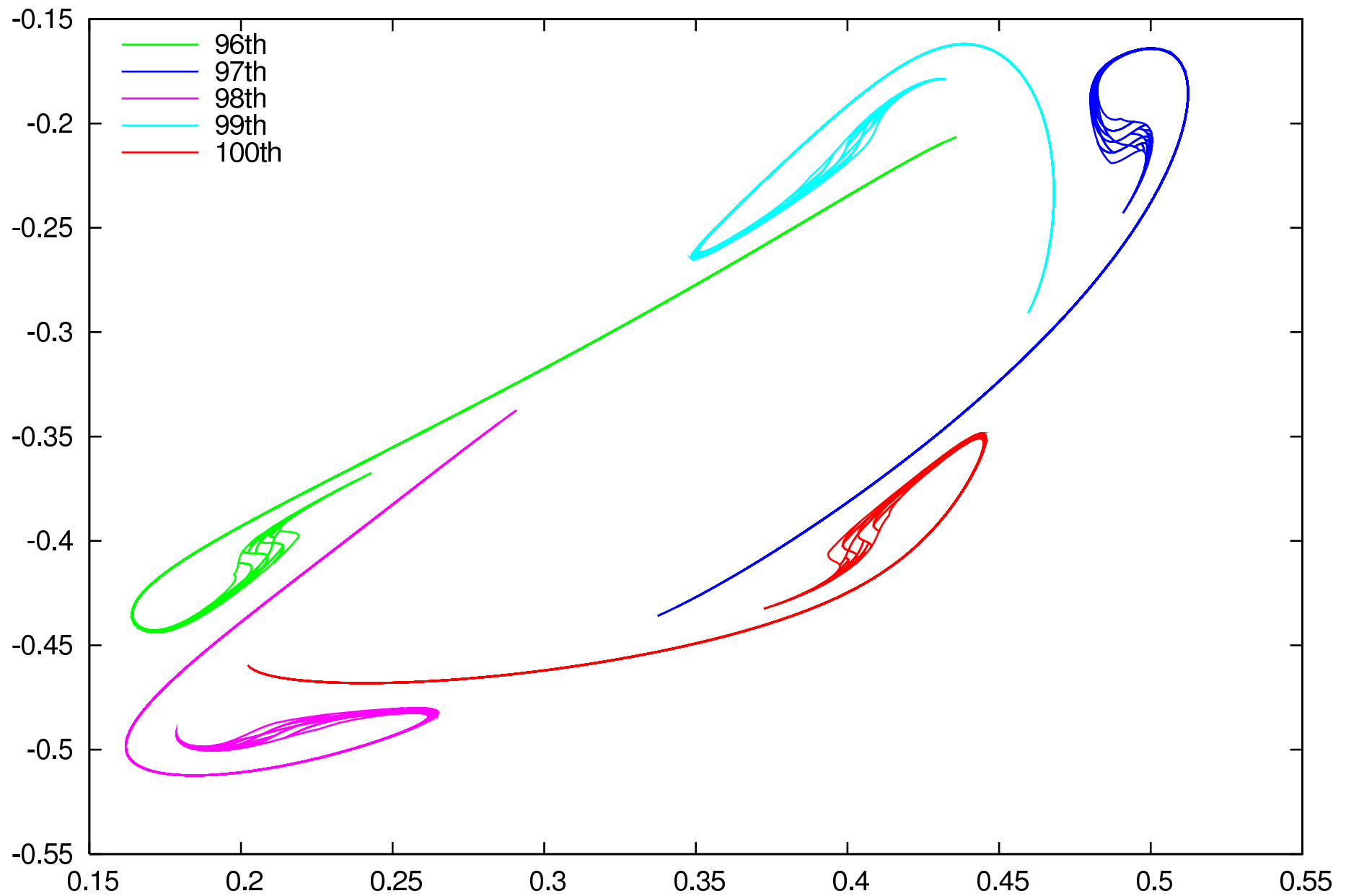
Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17



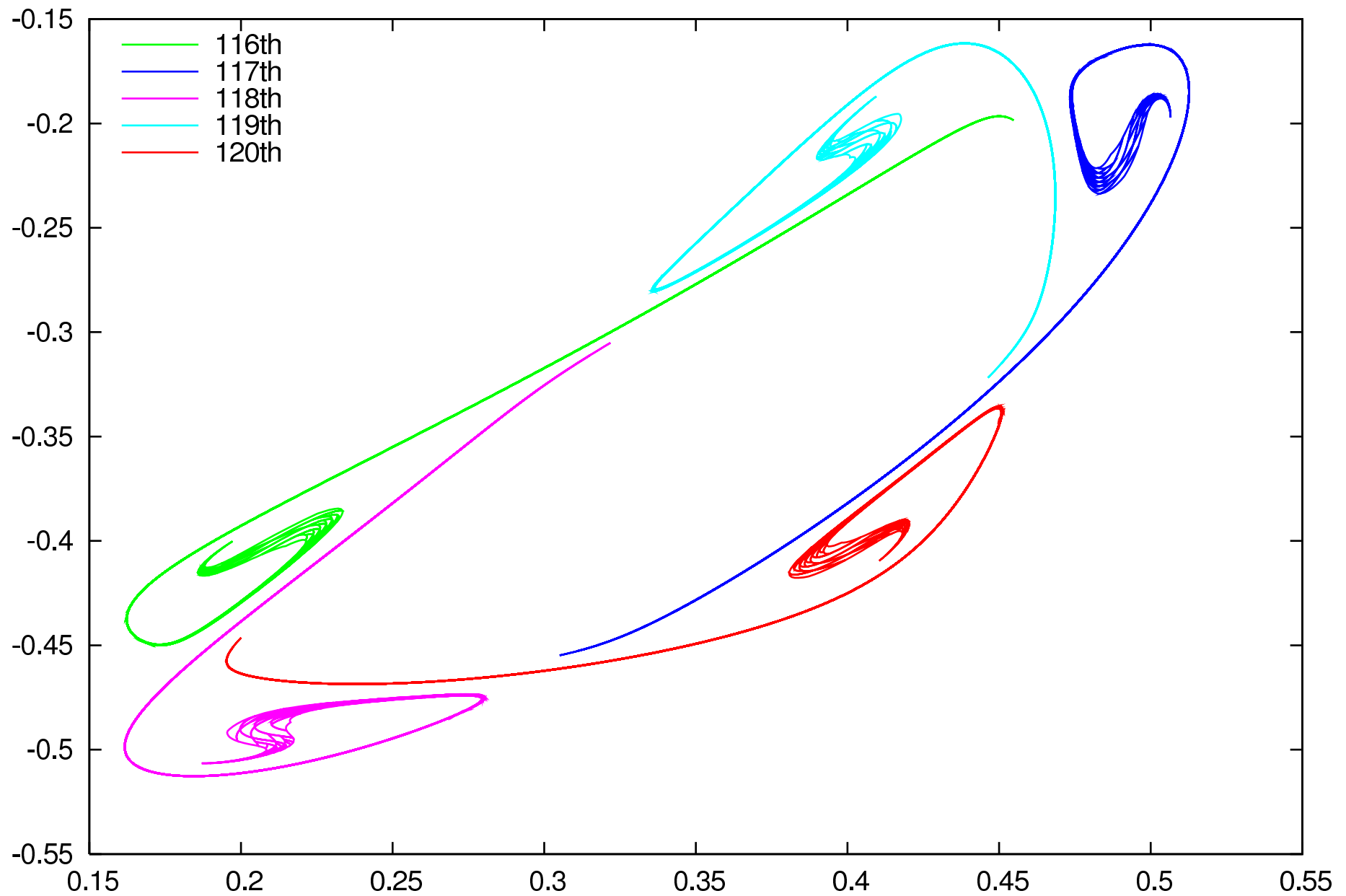
Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17

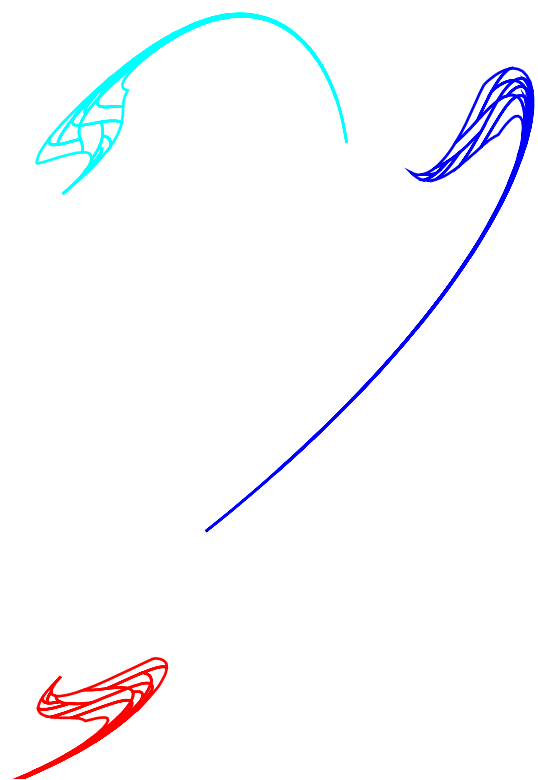


Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17

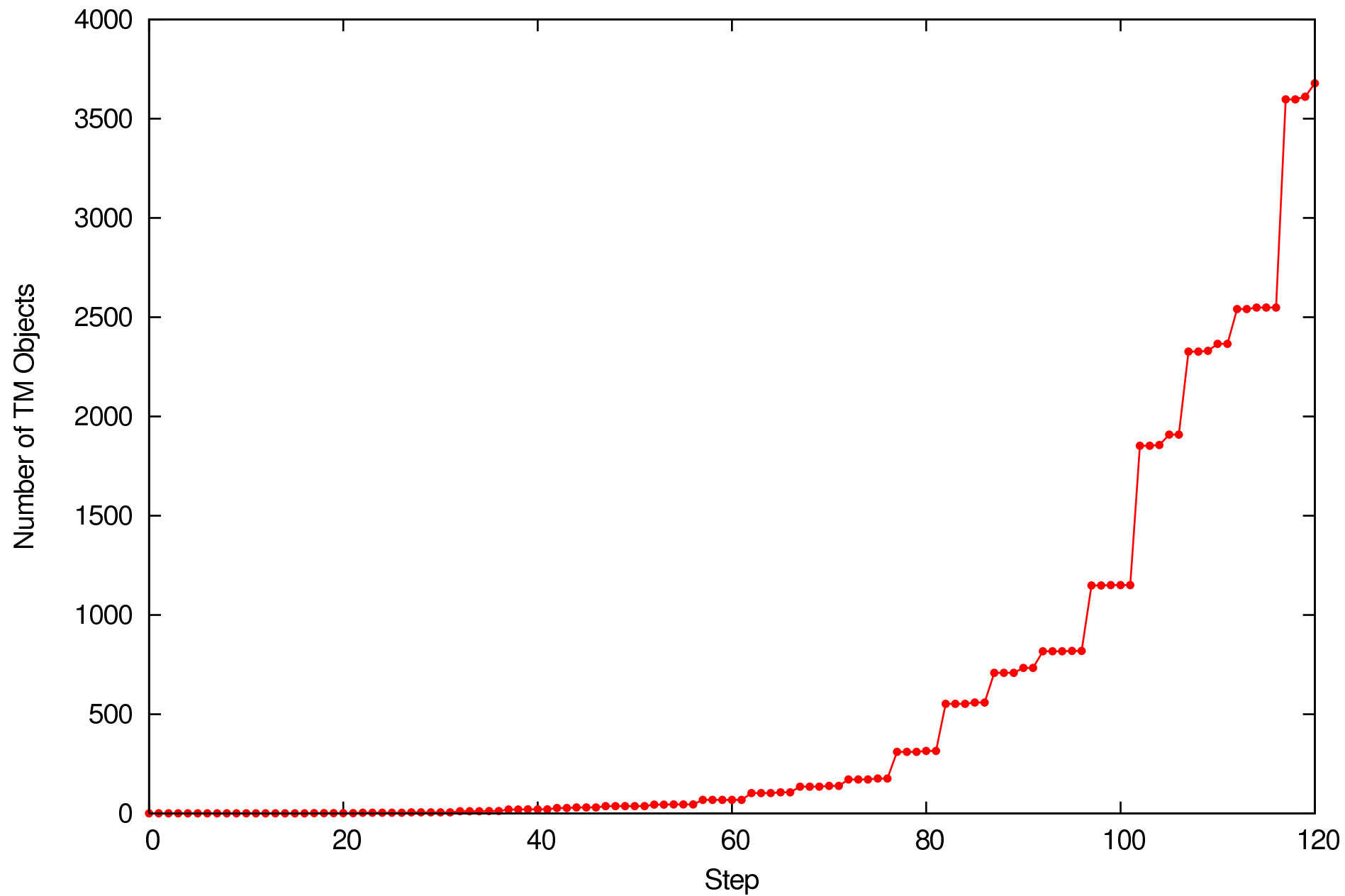


Henon system,  $x_n = 1 - 2.4x^2 + y$ ,  $y_n = -x$ , NO=33 w17

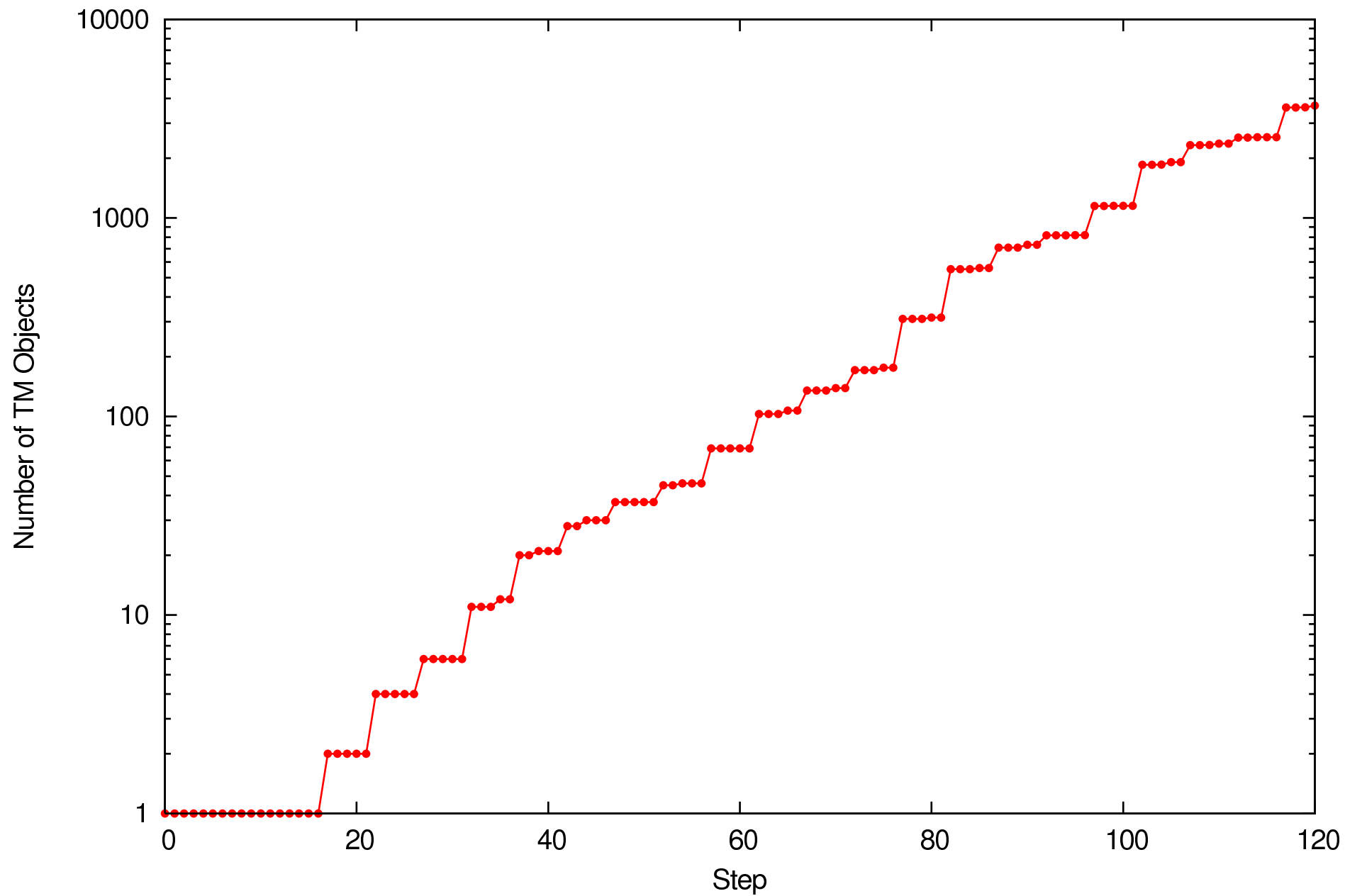




henonL: Count of TM Objects, NO=33, Psum0.5, all P splits (e-10,2coins)

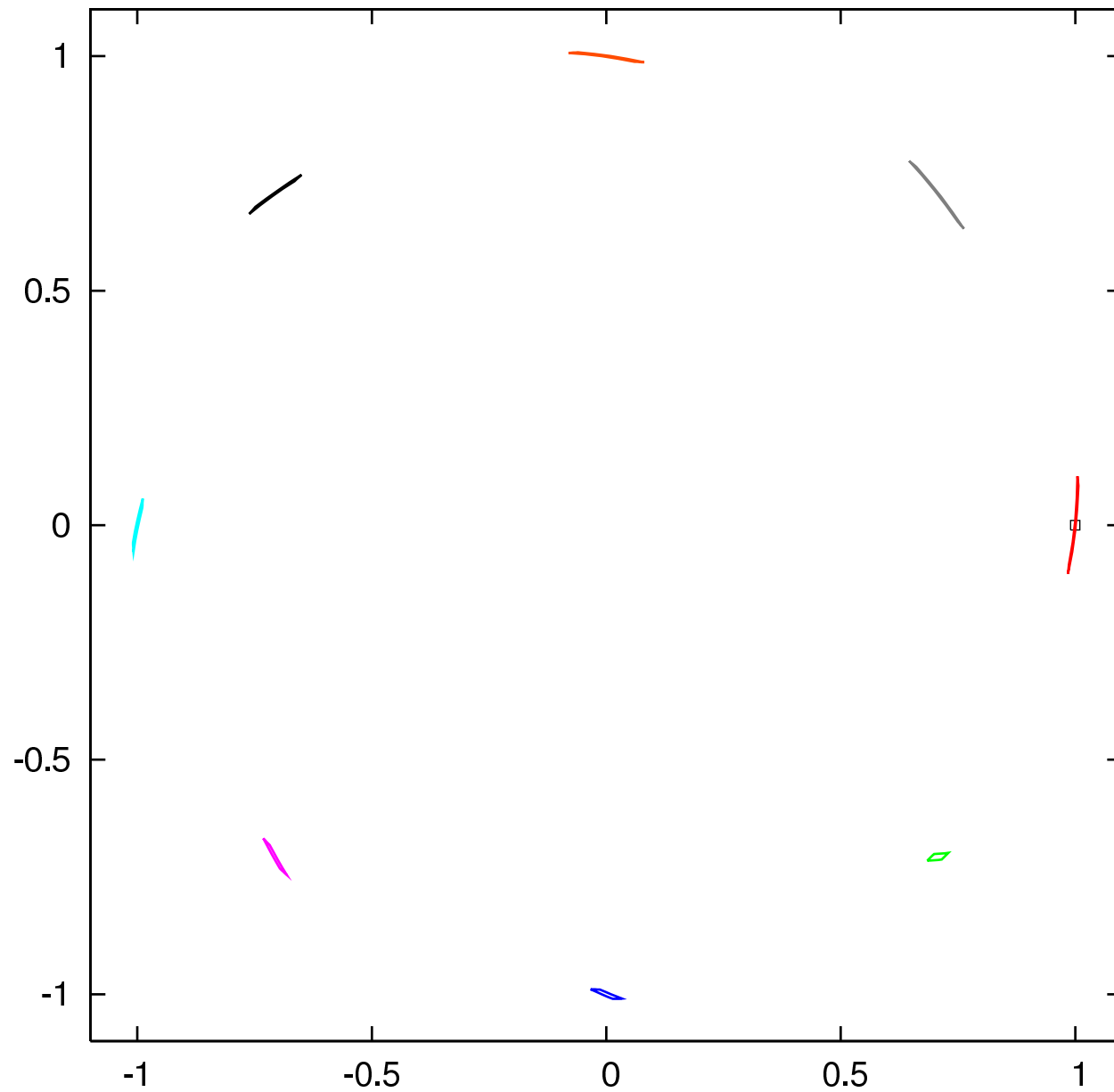


henonL: Count of TM Objects, NO=33, Psum0.5, all P splits (e-10,2coins)

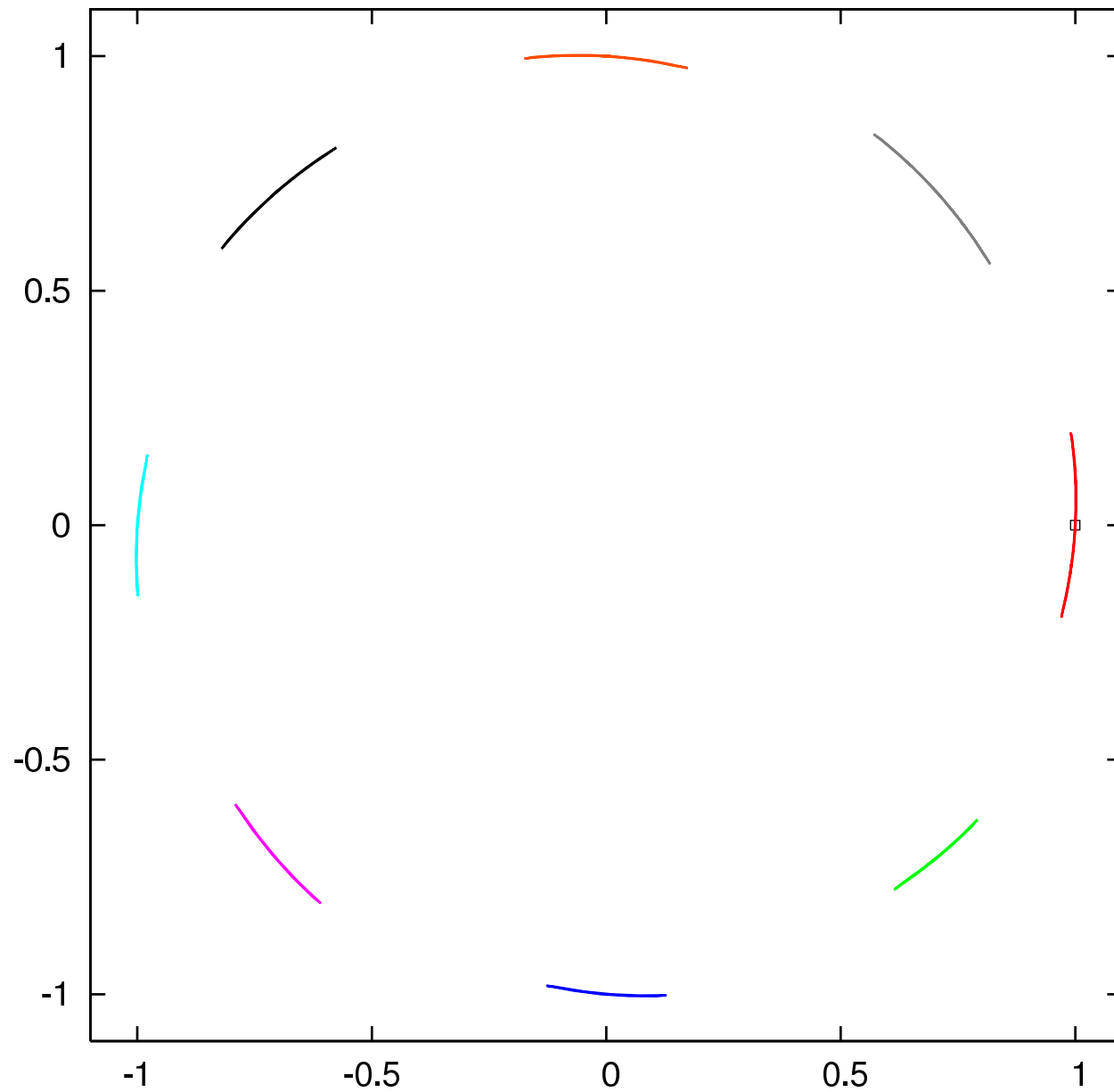




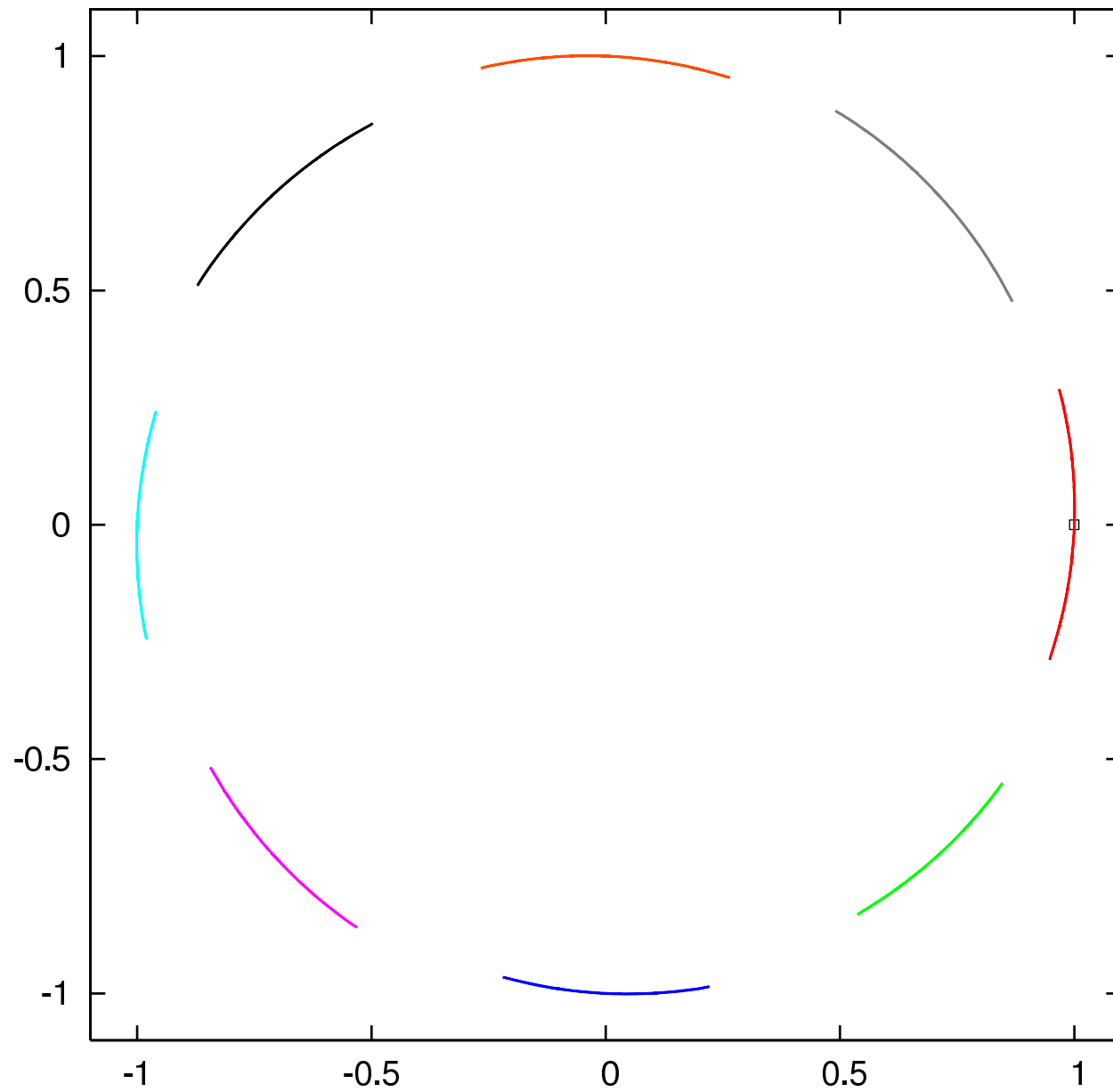
**discrete kepler. 1st revolution, ICw 0.02, NO=13 w7**



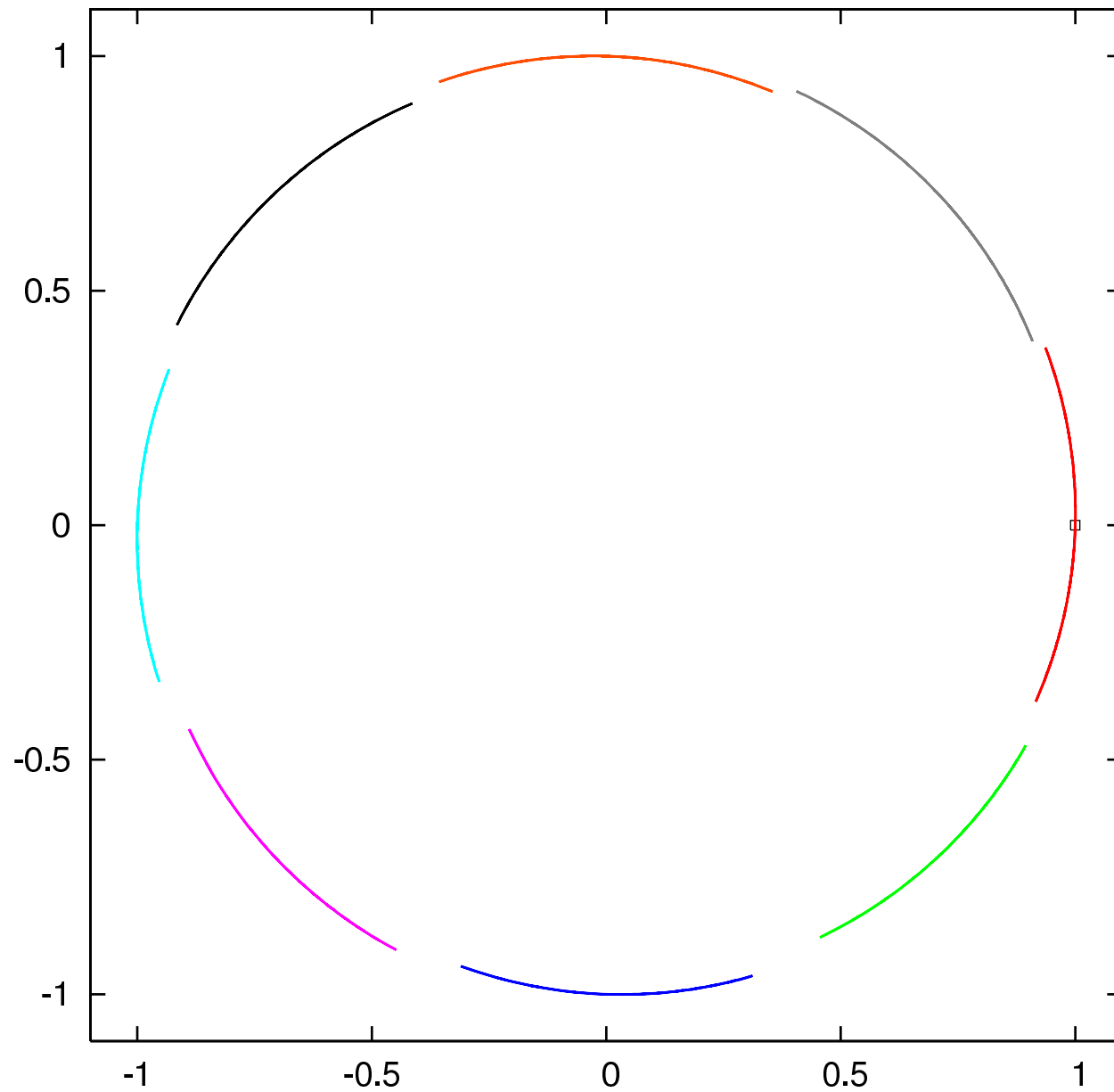
**discrete kepler. 2nd revolution, ICw 0.02, NO=13 w7**



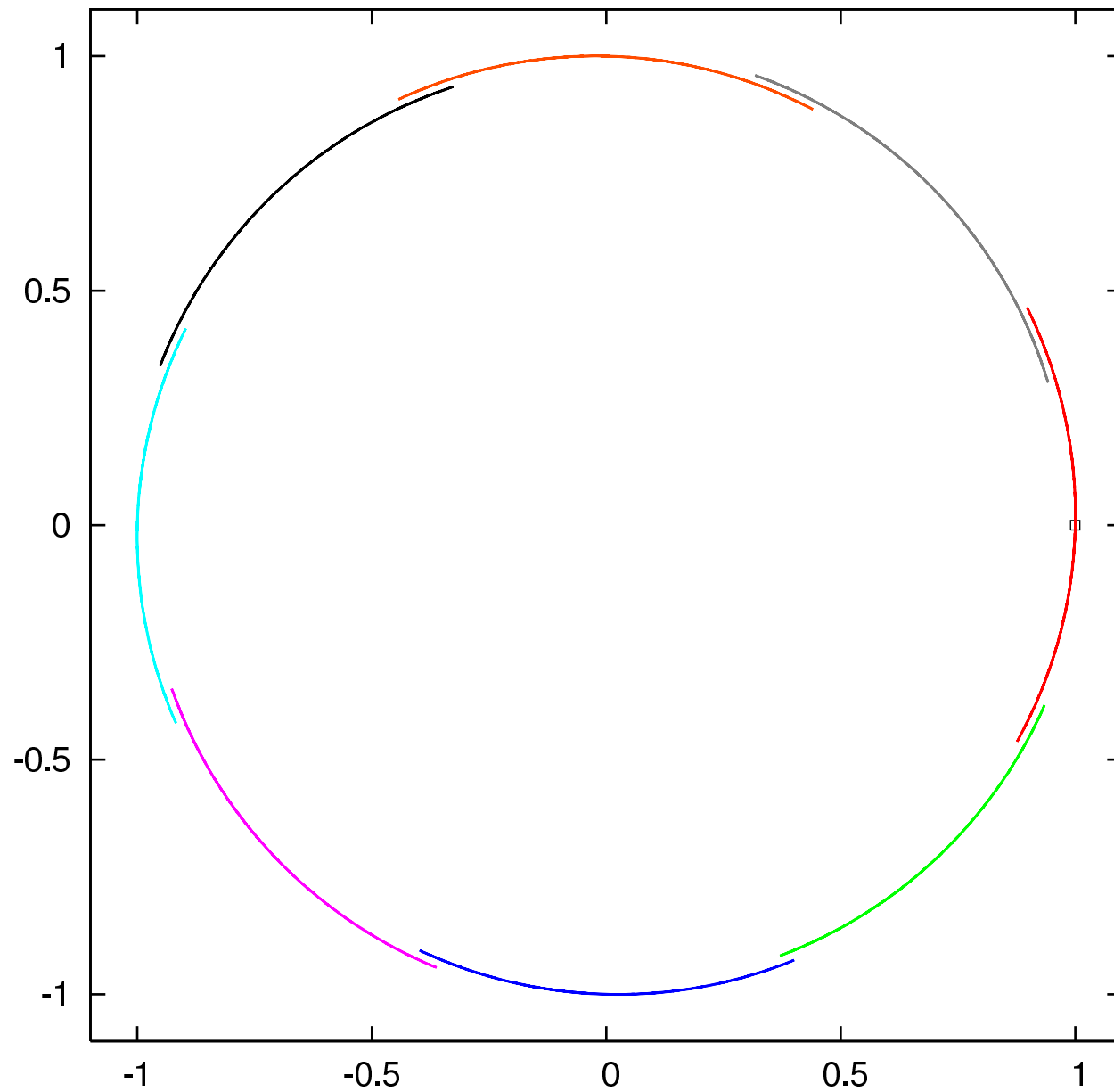
**discrete kepler. 3rd revolution, ICw 0.02, NO=13 w7**



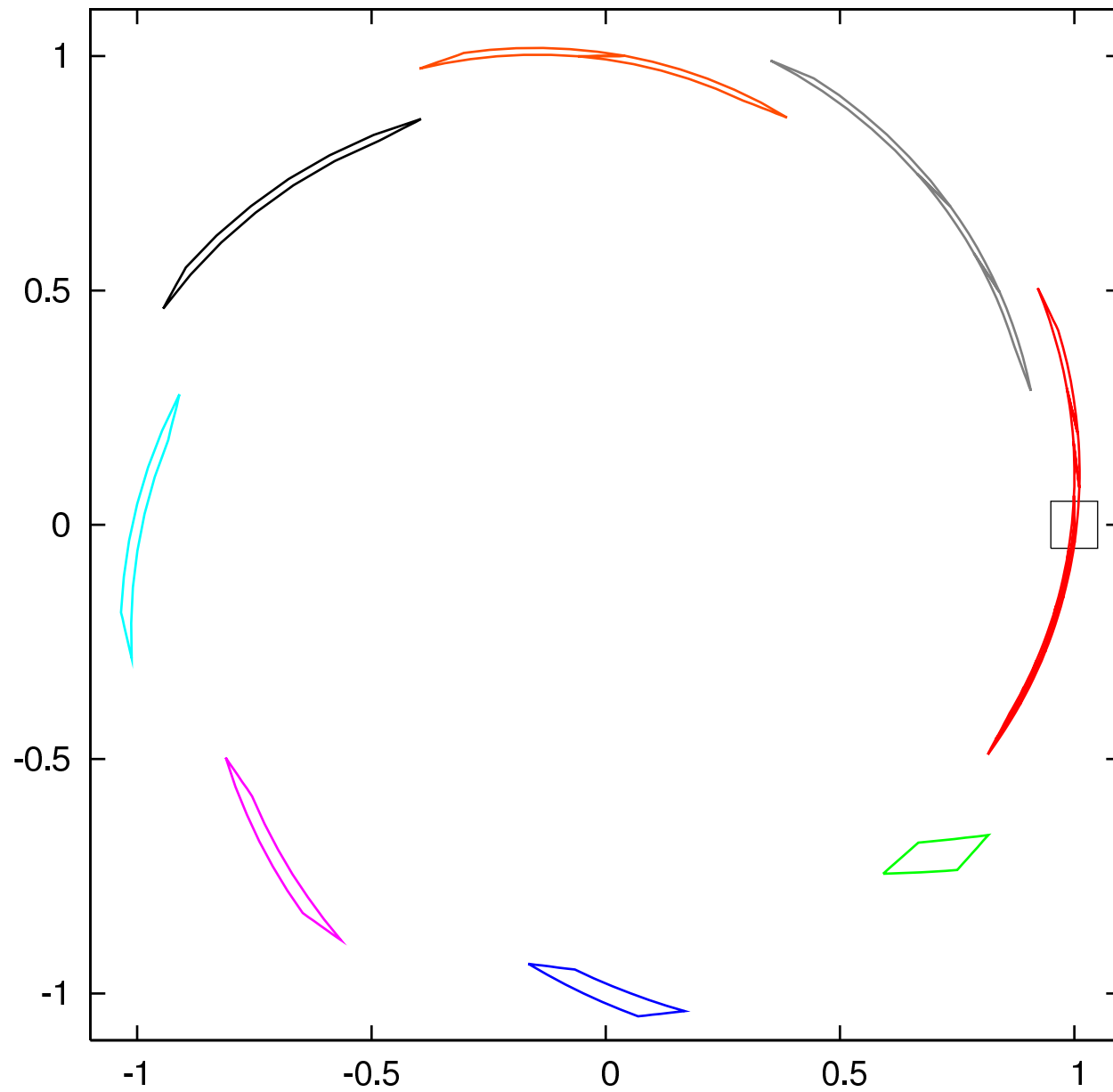
**discrete kepler. 4th revolution, ICw 0.02, NO=13 w7**



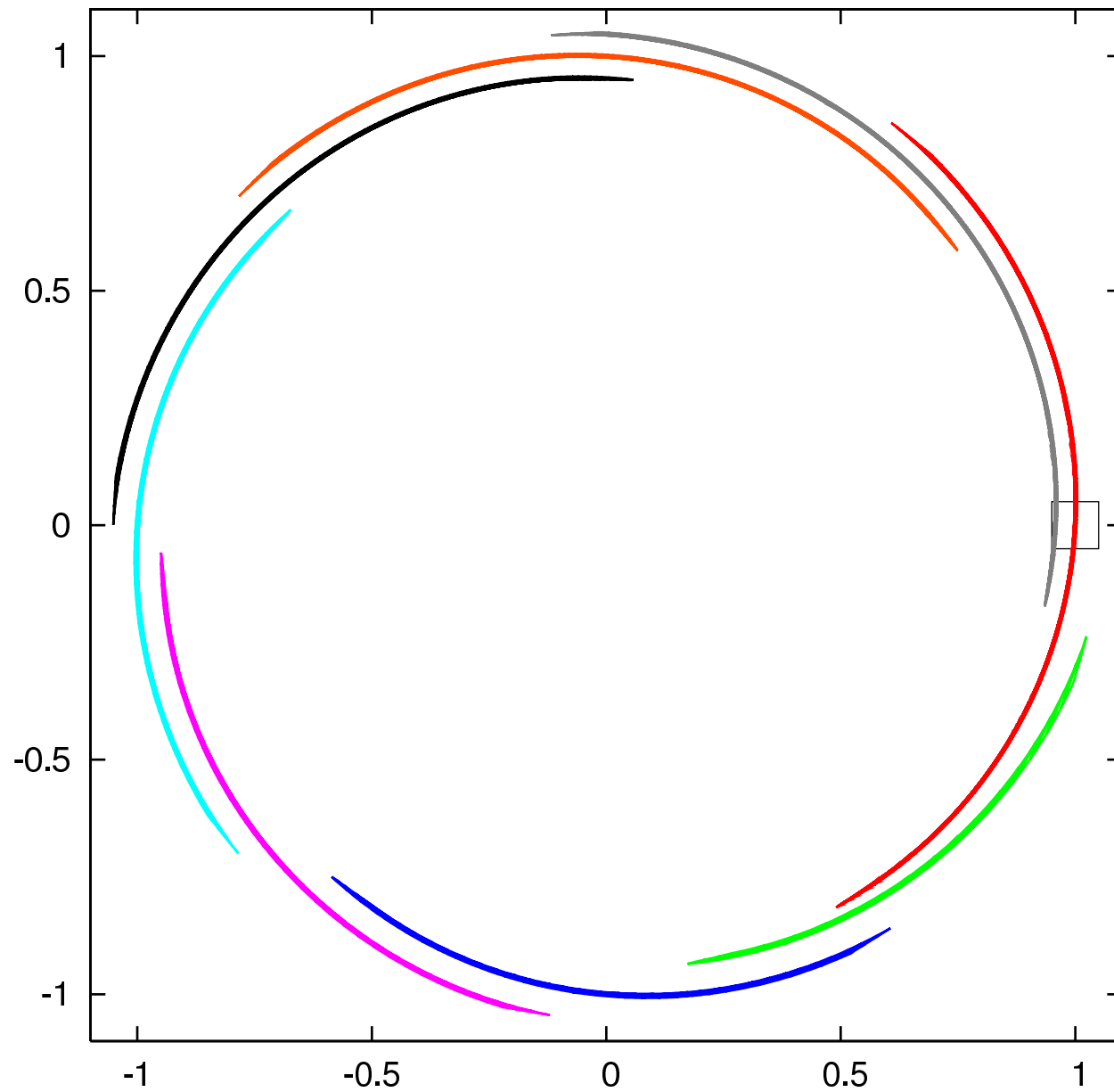
**discrete kepler. 5th revolution, ICw 0.02, NO=13 w7**



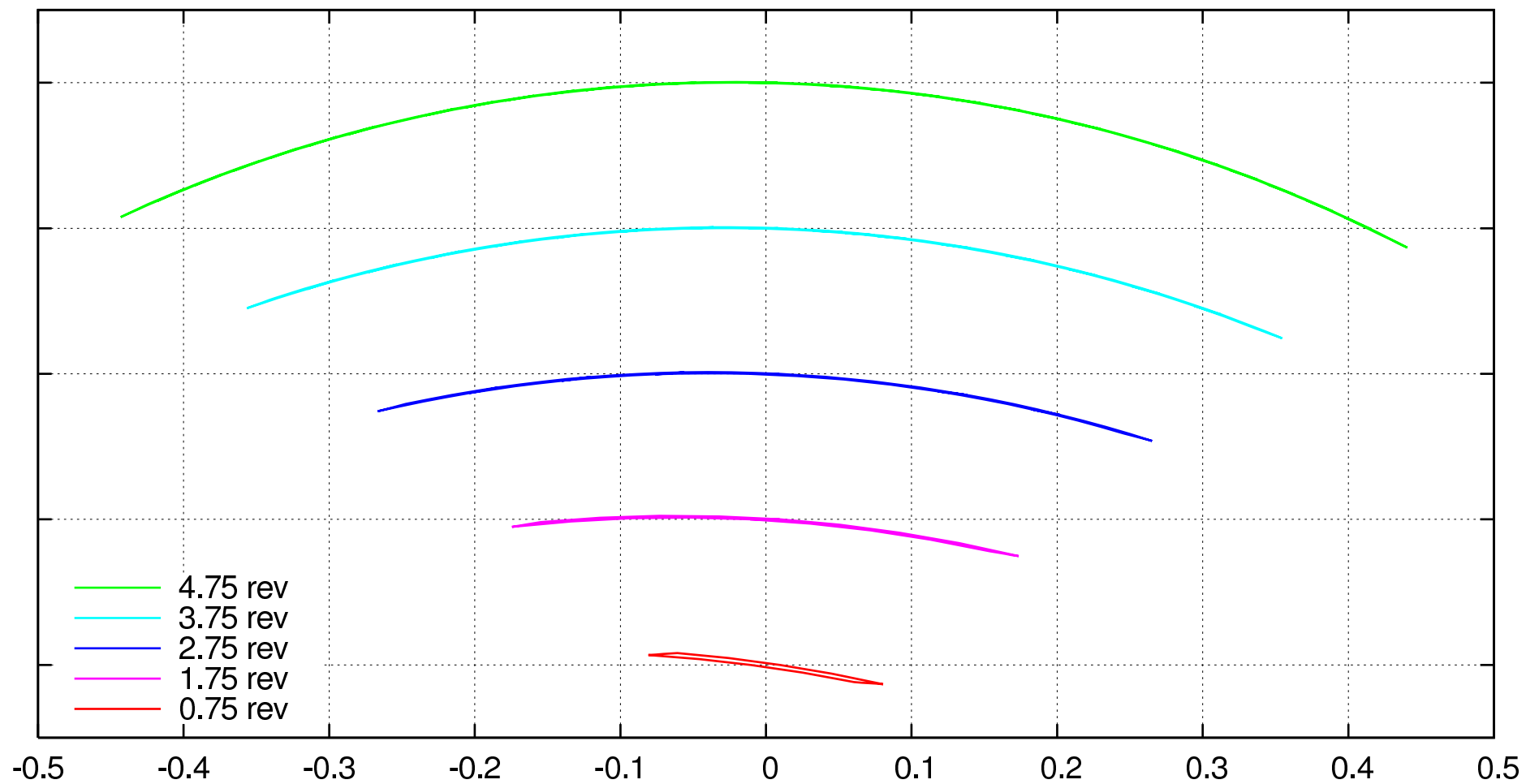
**discrete kepler. 1st revolution, ICw 0.1, NO=13 w7**



**discrete kepler. 2nd revolution, ICw 0.1, NO=13 w7**

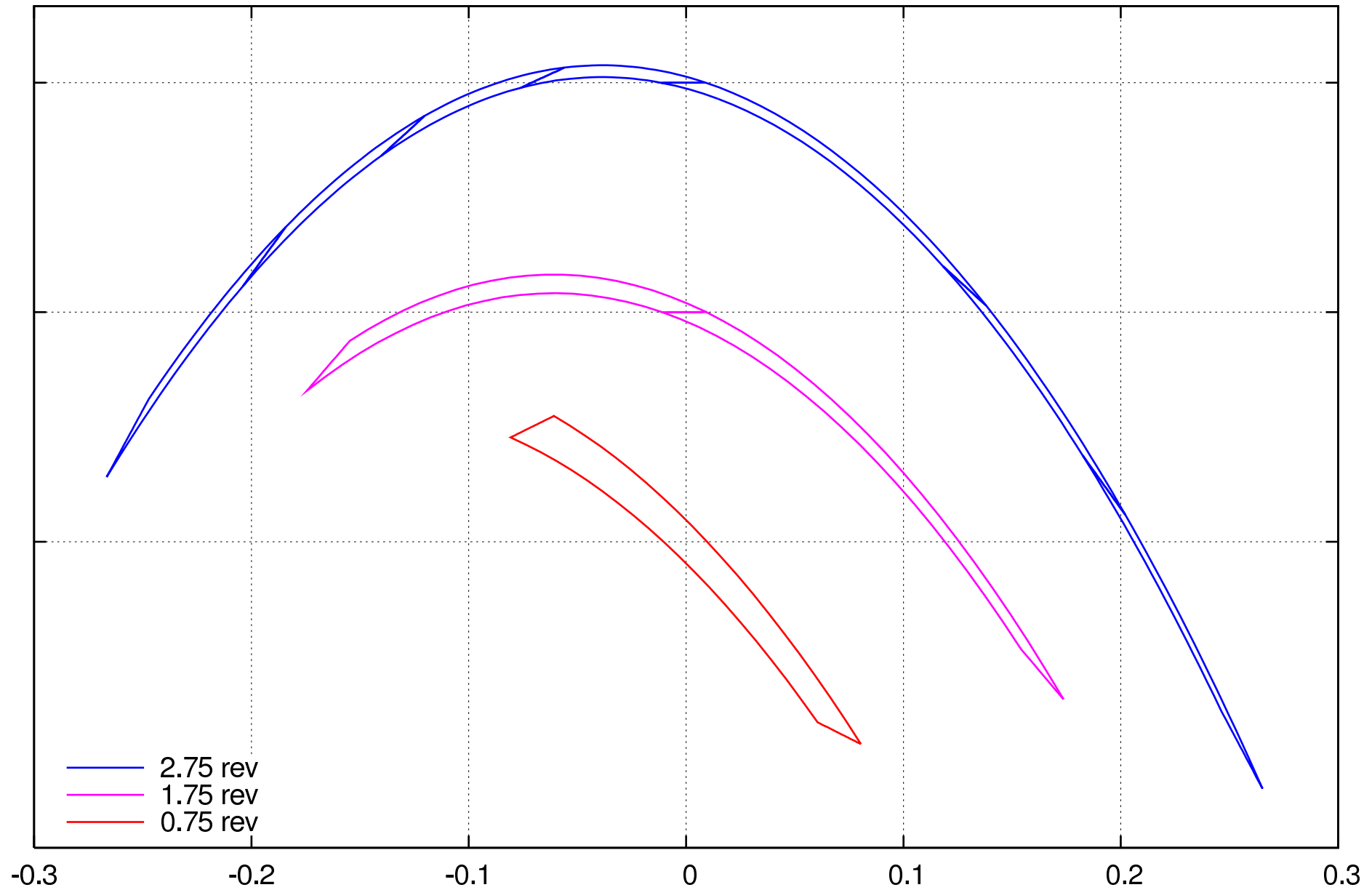


# discrete kepler. NO=13 w7

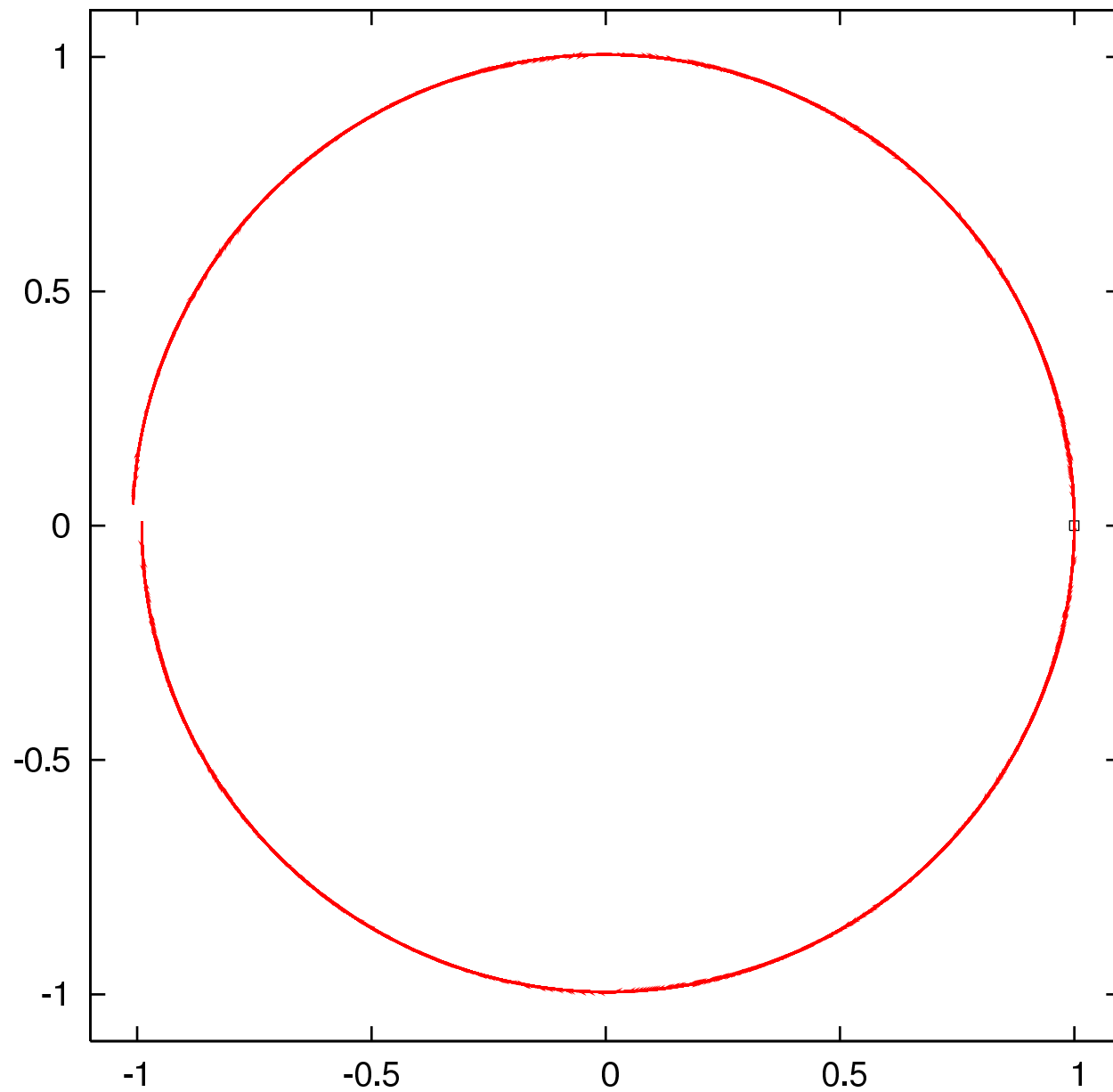




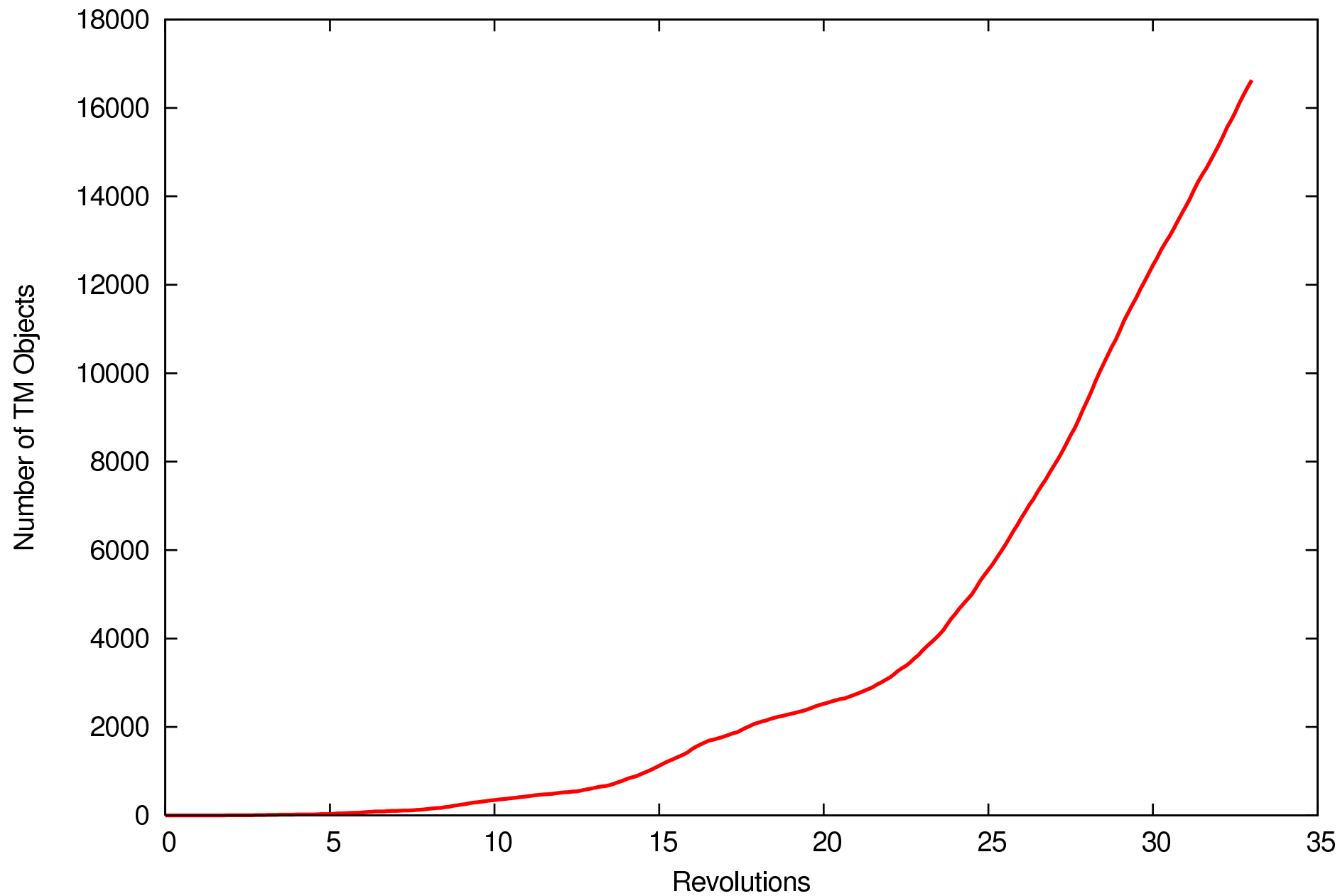
# discrete kepler. NO=13 w7



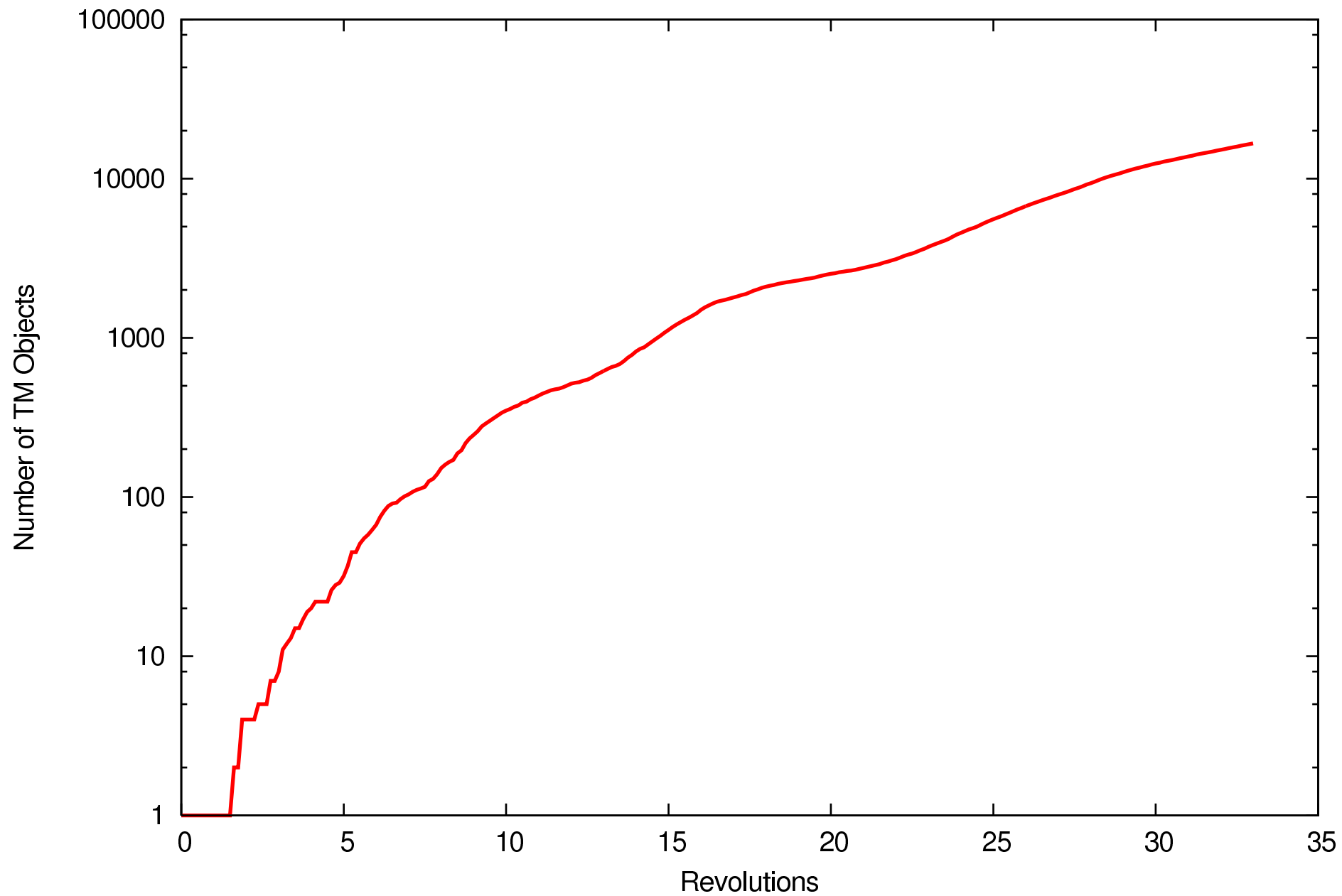
**discrete kepler. 33rd revolution, ICw 0.02, NO=13 w7**



discrete kepler: Count of TM Objects, ICw 0.02, NO=13, Psum0.5, all P splits (e-10,2coins)



discrete kepler: Count of TM Objects, ICw 0.02, NO=13, Psum0.5, all P splits (e-10,2coins)



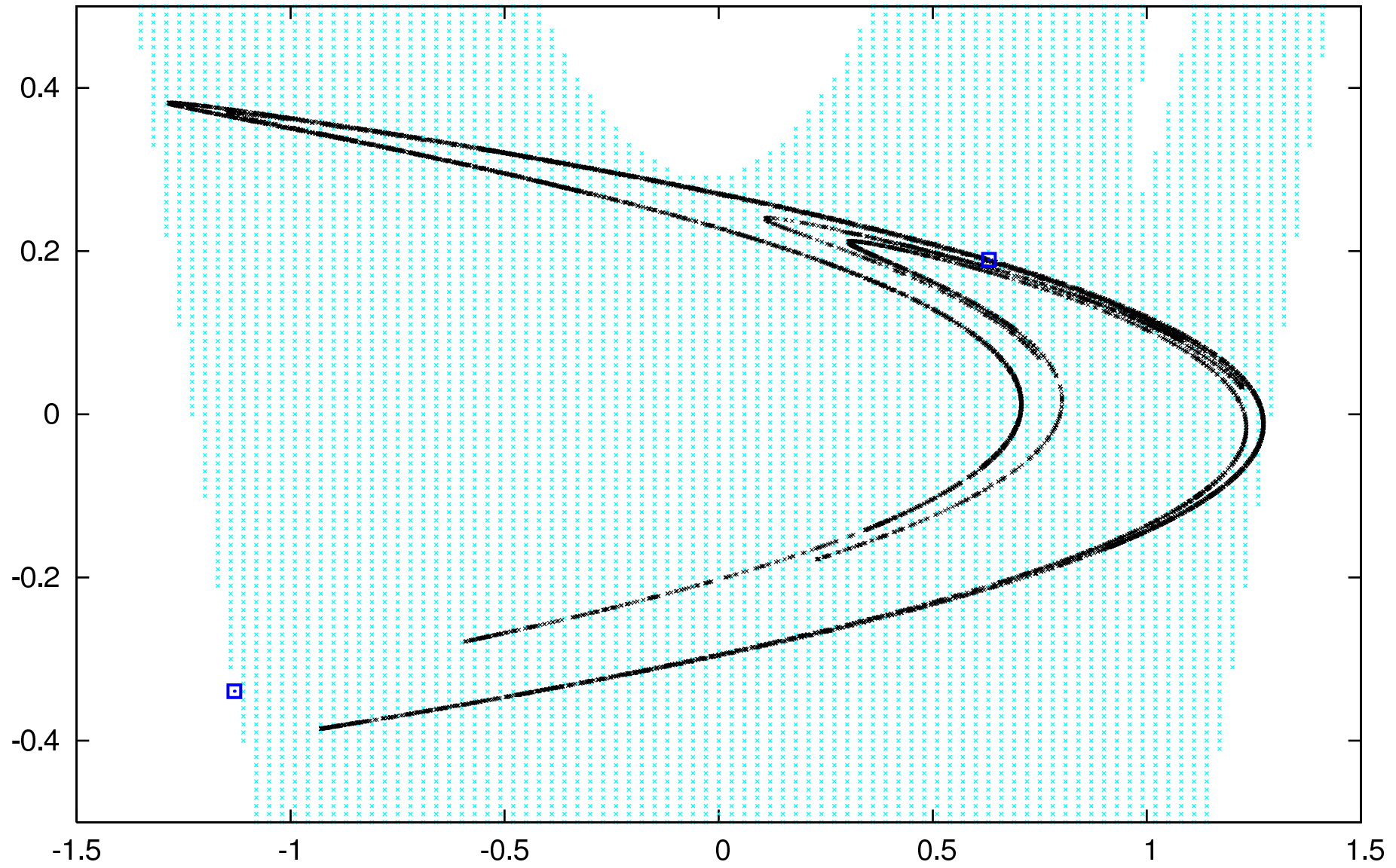
# The Henon Map

$$H(x, y) = (1 - ax^2 + y, bx).$$

We set the parameters  $a = 1.4$  and  $b = 0.3$ , which are originally considered by Henon. The map  $H$  has two fixed points.

$$\vec{p}_1 = (0.63135, 0.18940) \quad \text{and} \quad \vec{p}_2 = (-1.13135, -0.33941).$$

rhonon. surviving region through 12 mappings



survived IC points

x

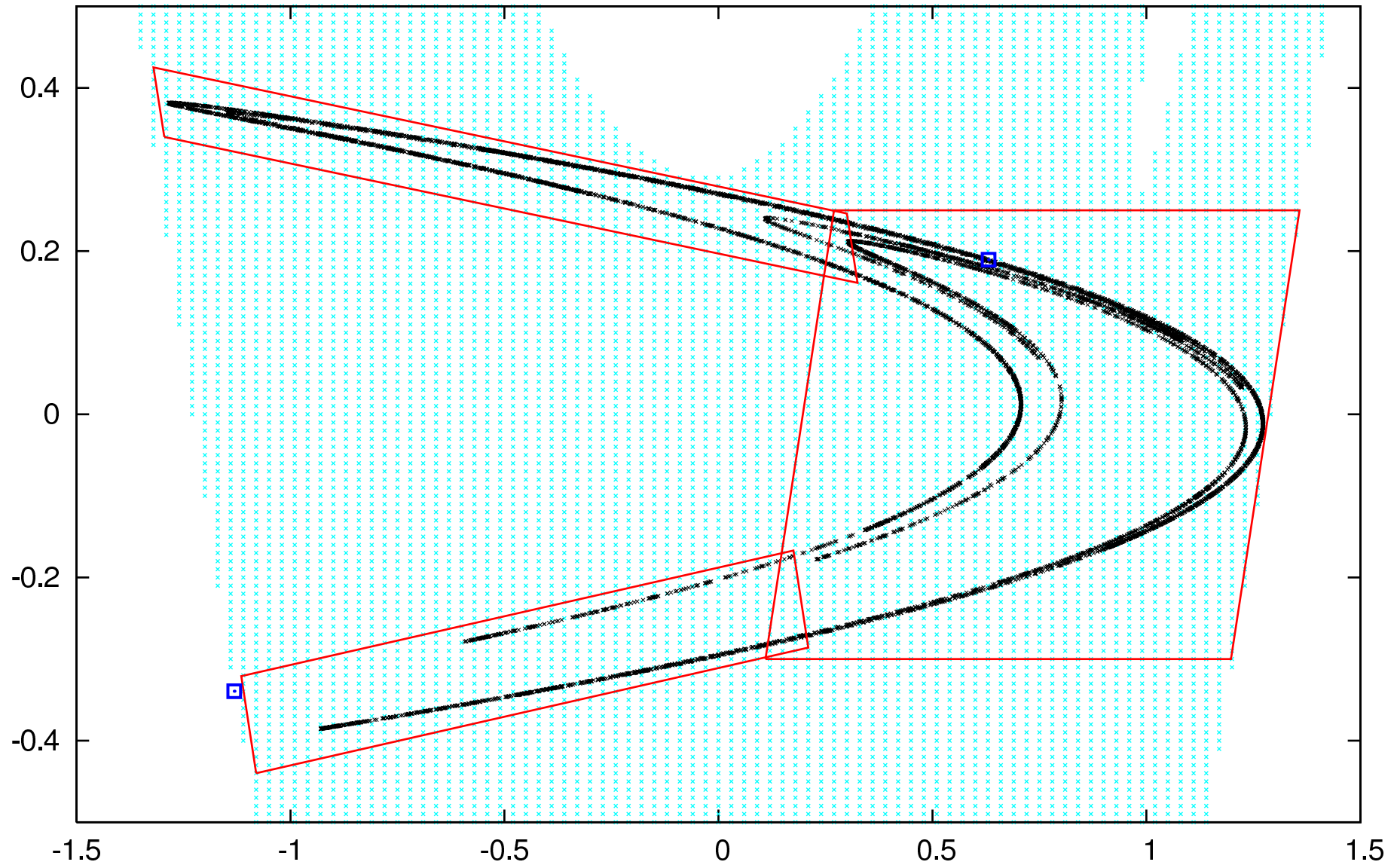
mapped points

x

fixed points

■

renon. surviving region through 12 mappings



survived IC points

x

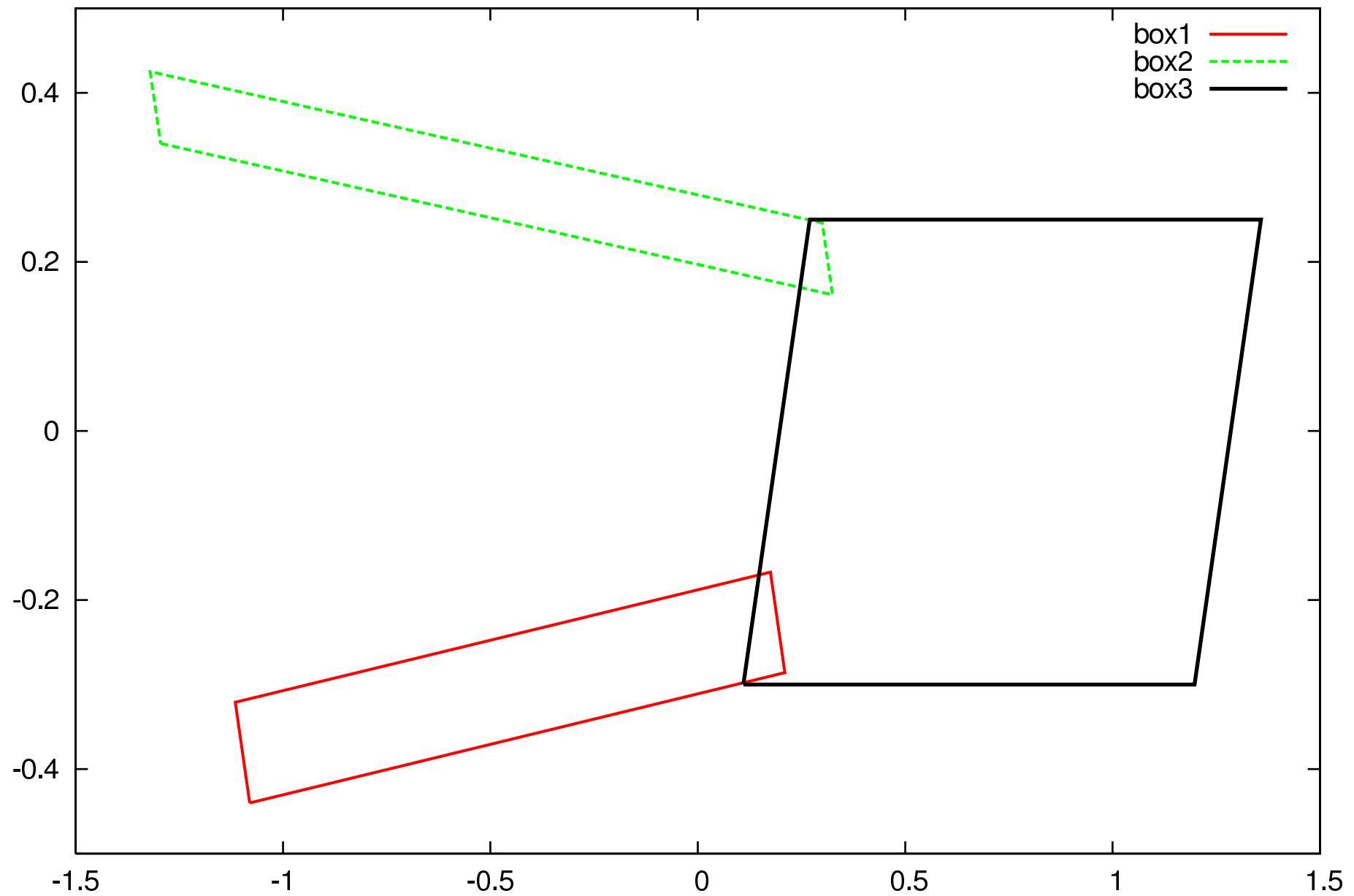
mapped points

x

fixed points

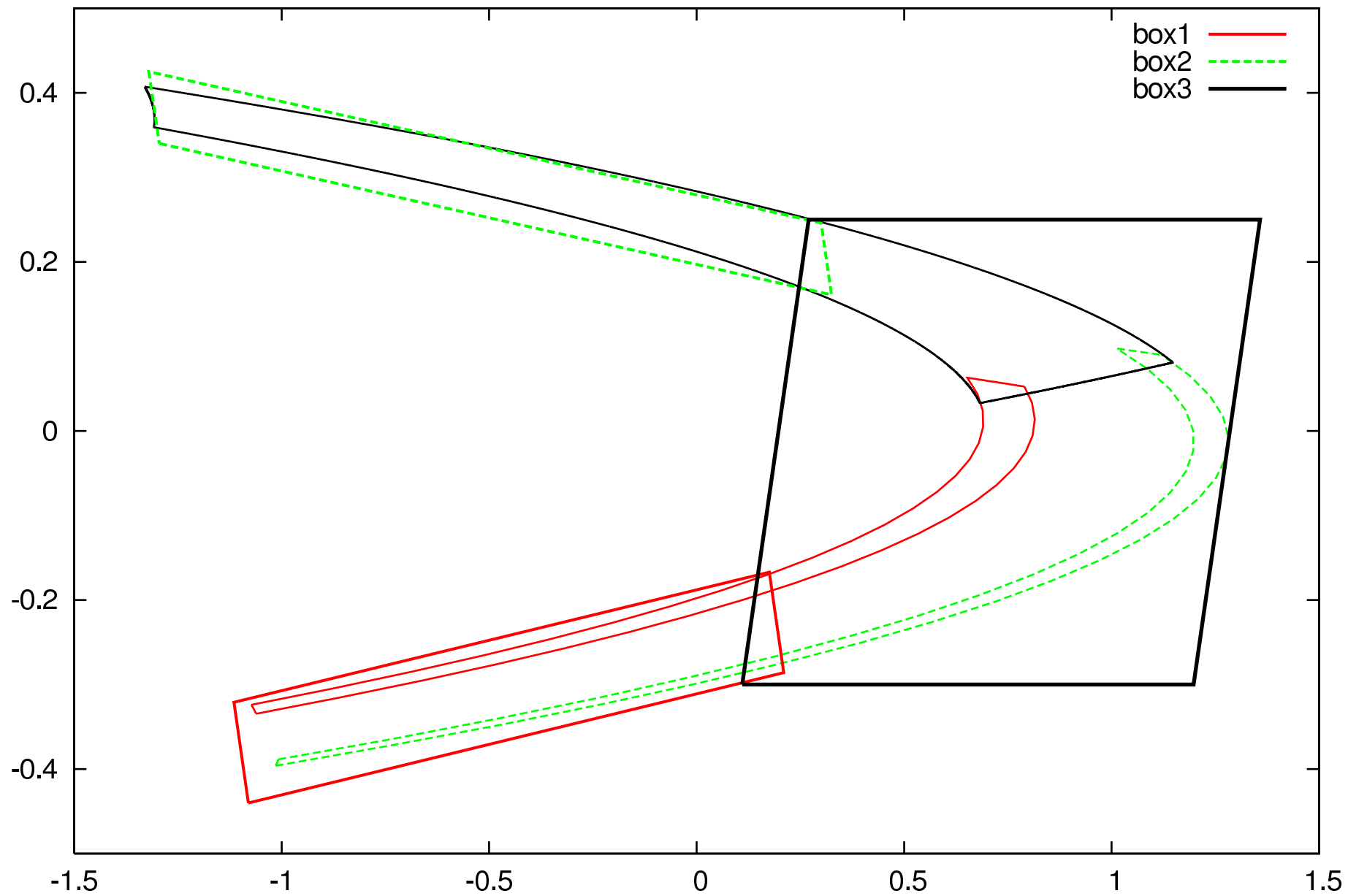
■

rhonon. IC boxes 3/3/08

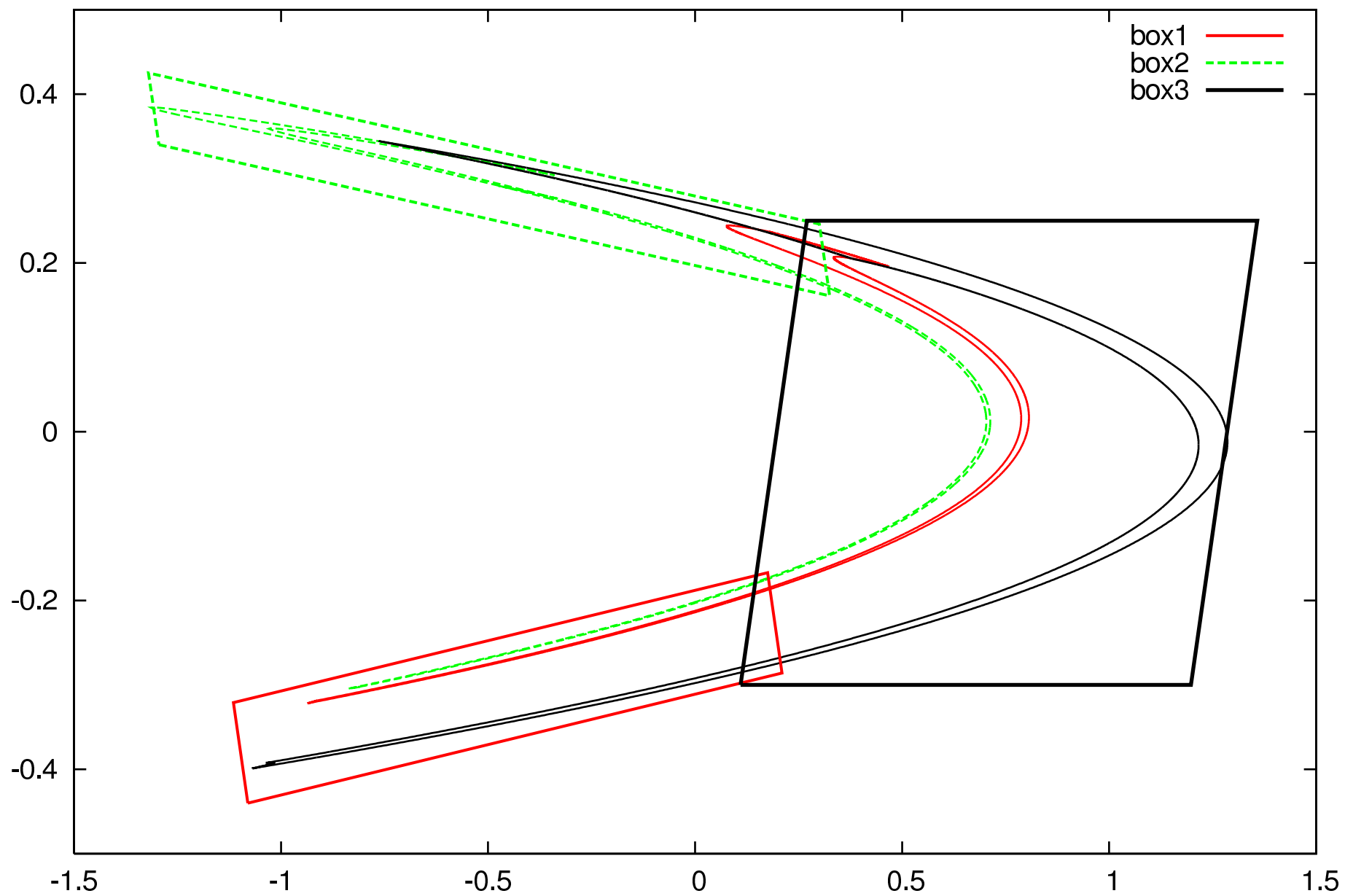




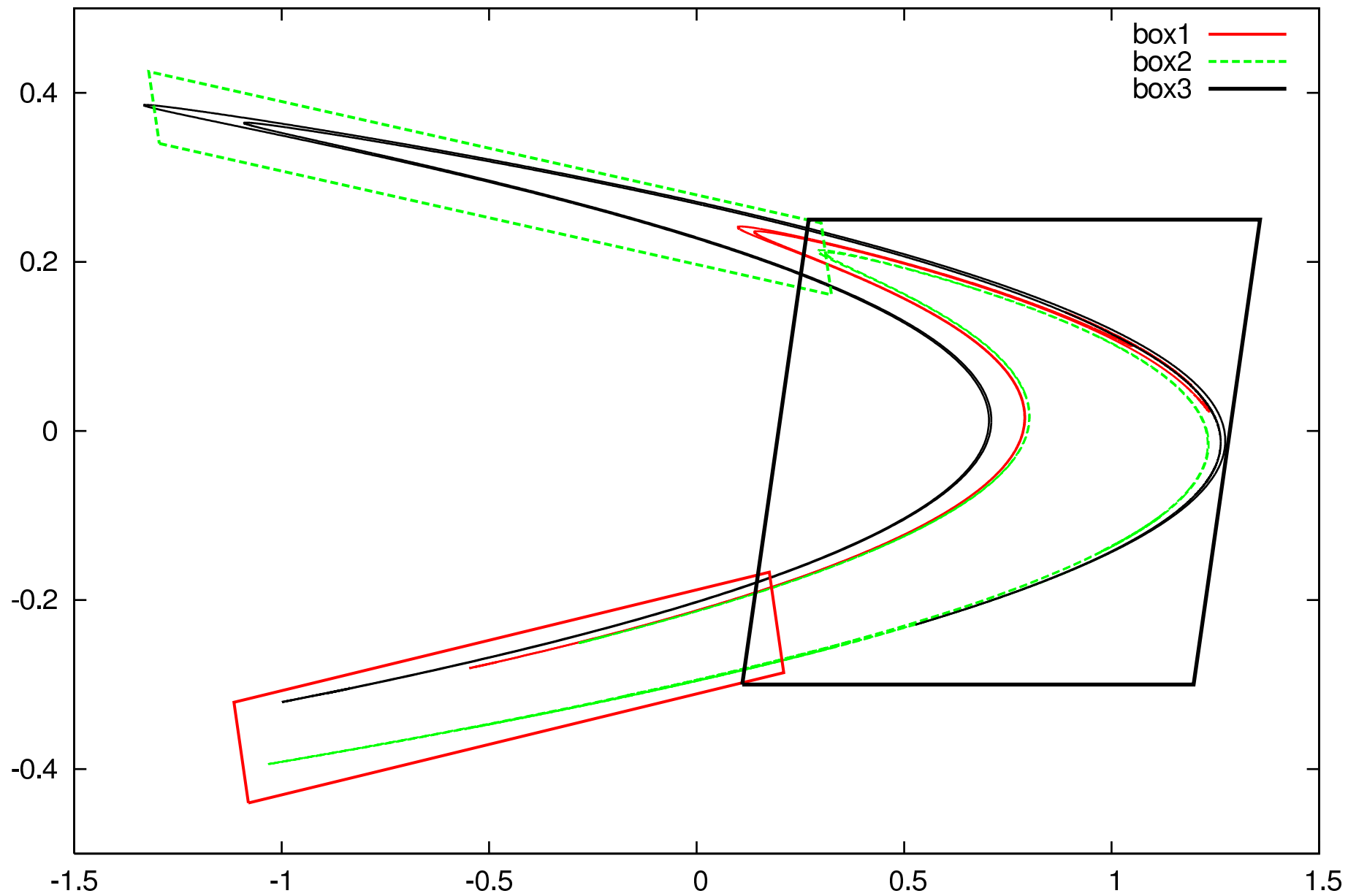
rhenon. step 1. 3/3/08



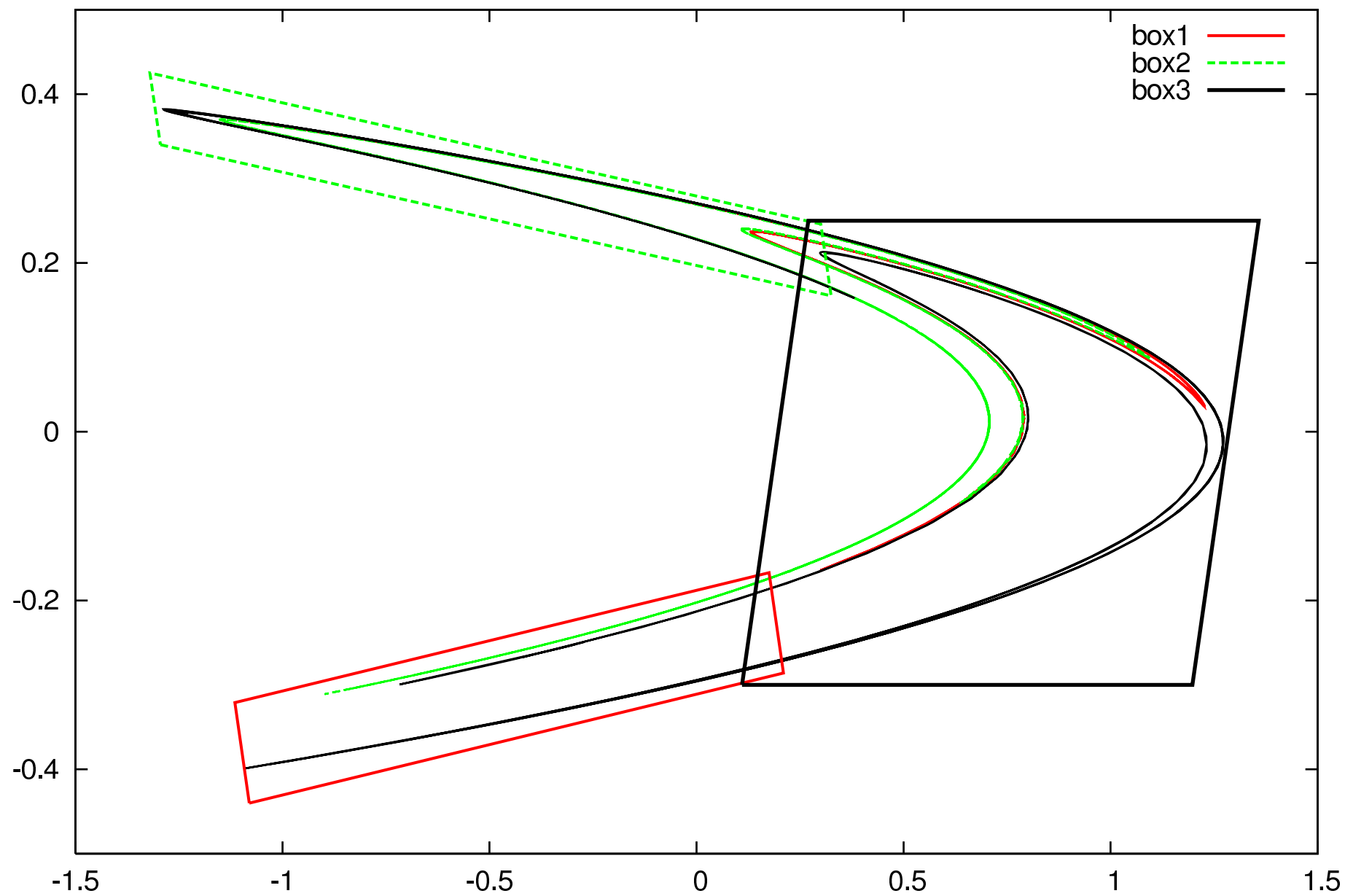
rhenon. step 2. 3/3/08



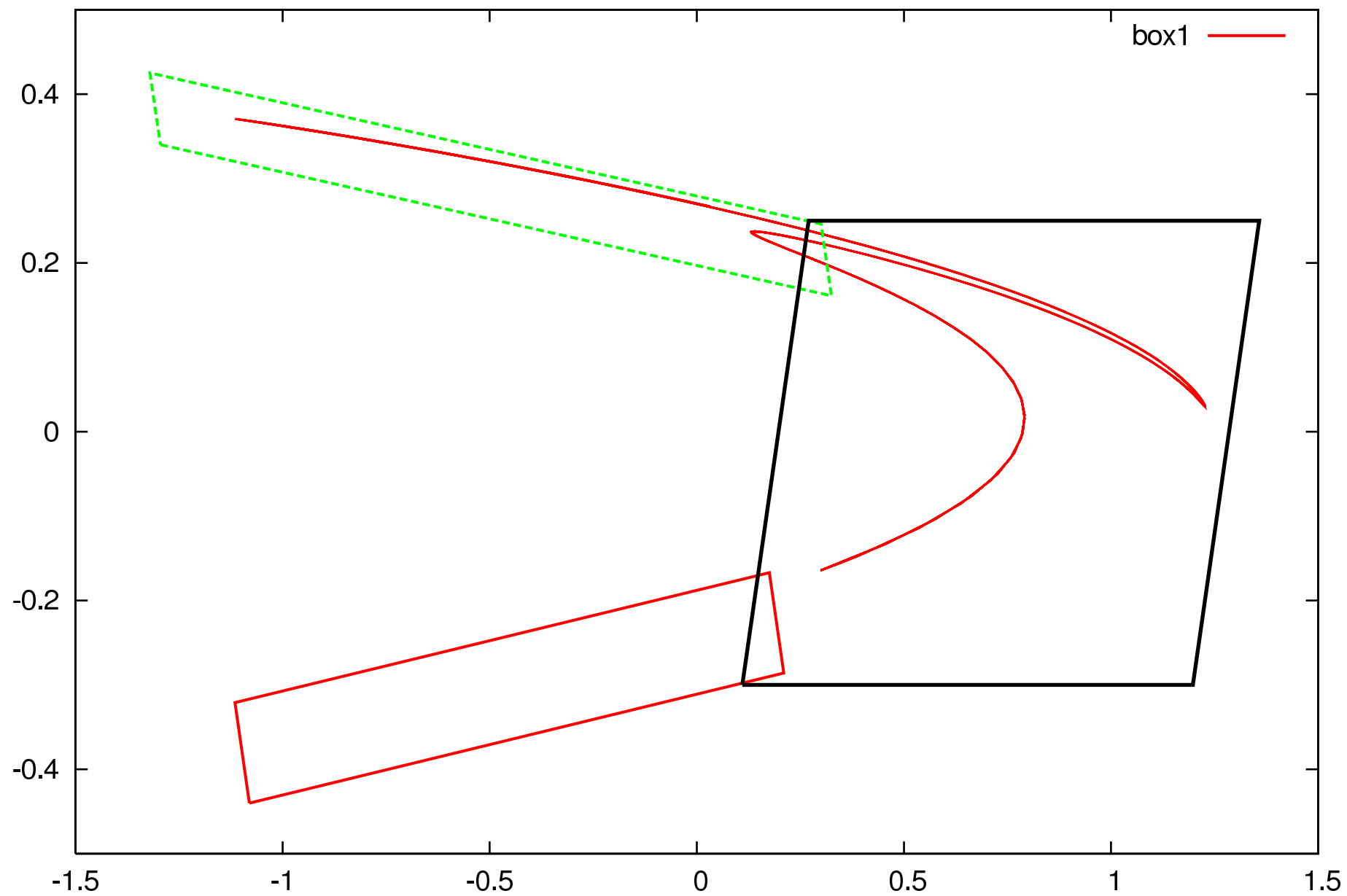
rhenon. step 3. 3/3/08



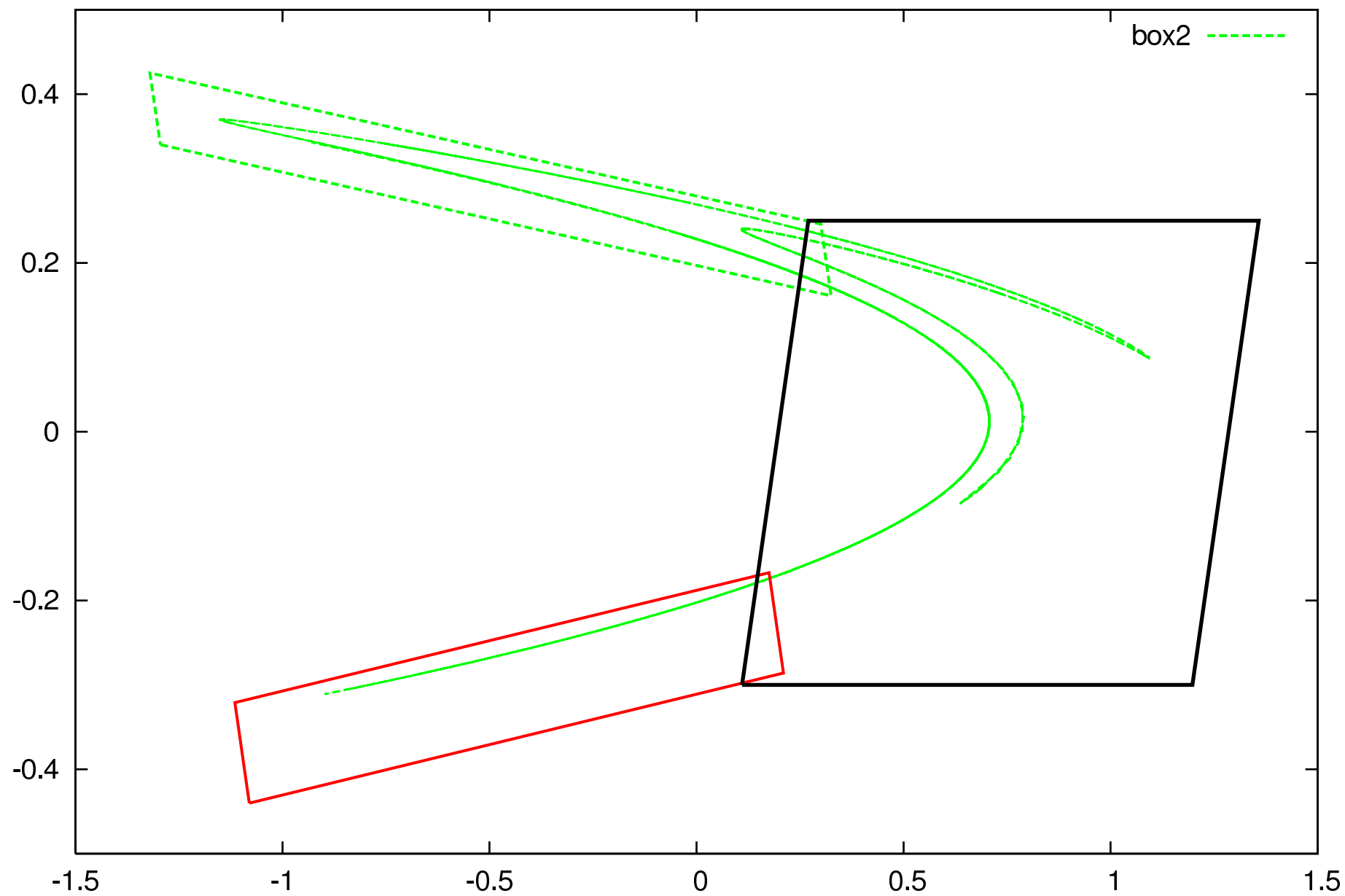
rhenon. step 4. 3/3/08



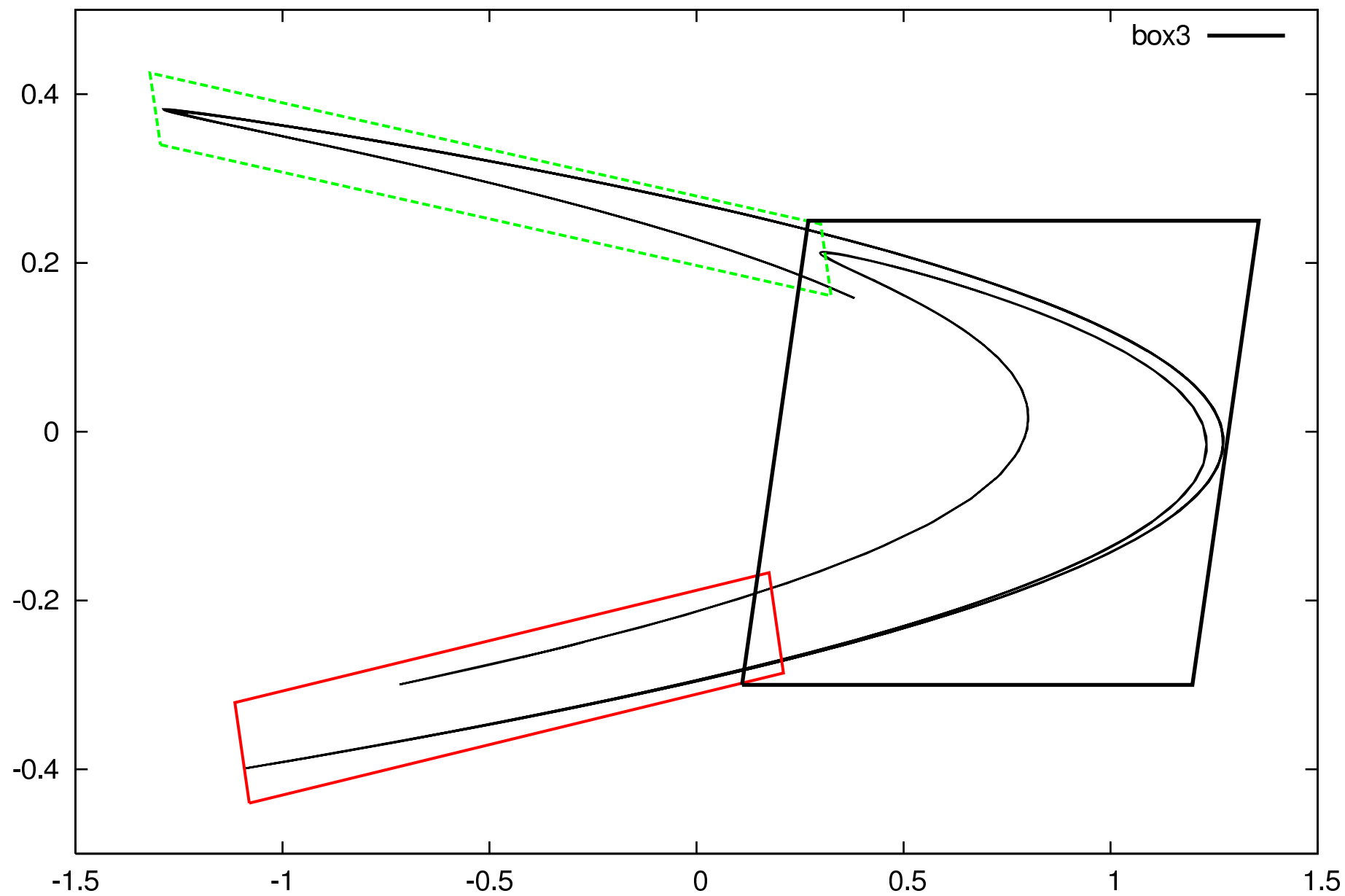
rhonon. step 4. box1. 3/3/08



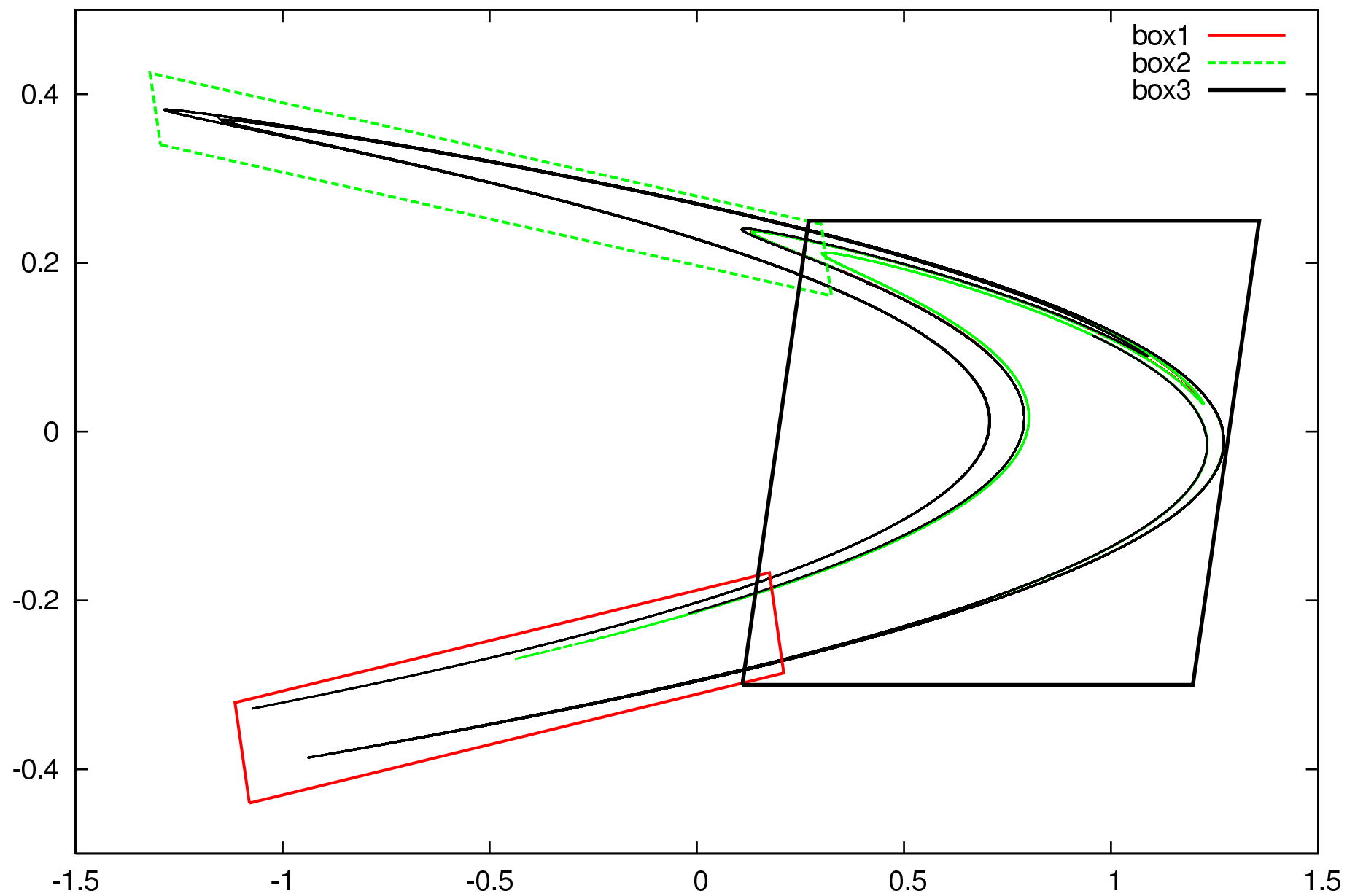
rhonon. step 4. box2. 3/3/08



rhonon. step 4. box3. 3/3/08

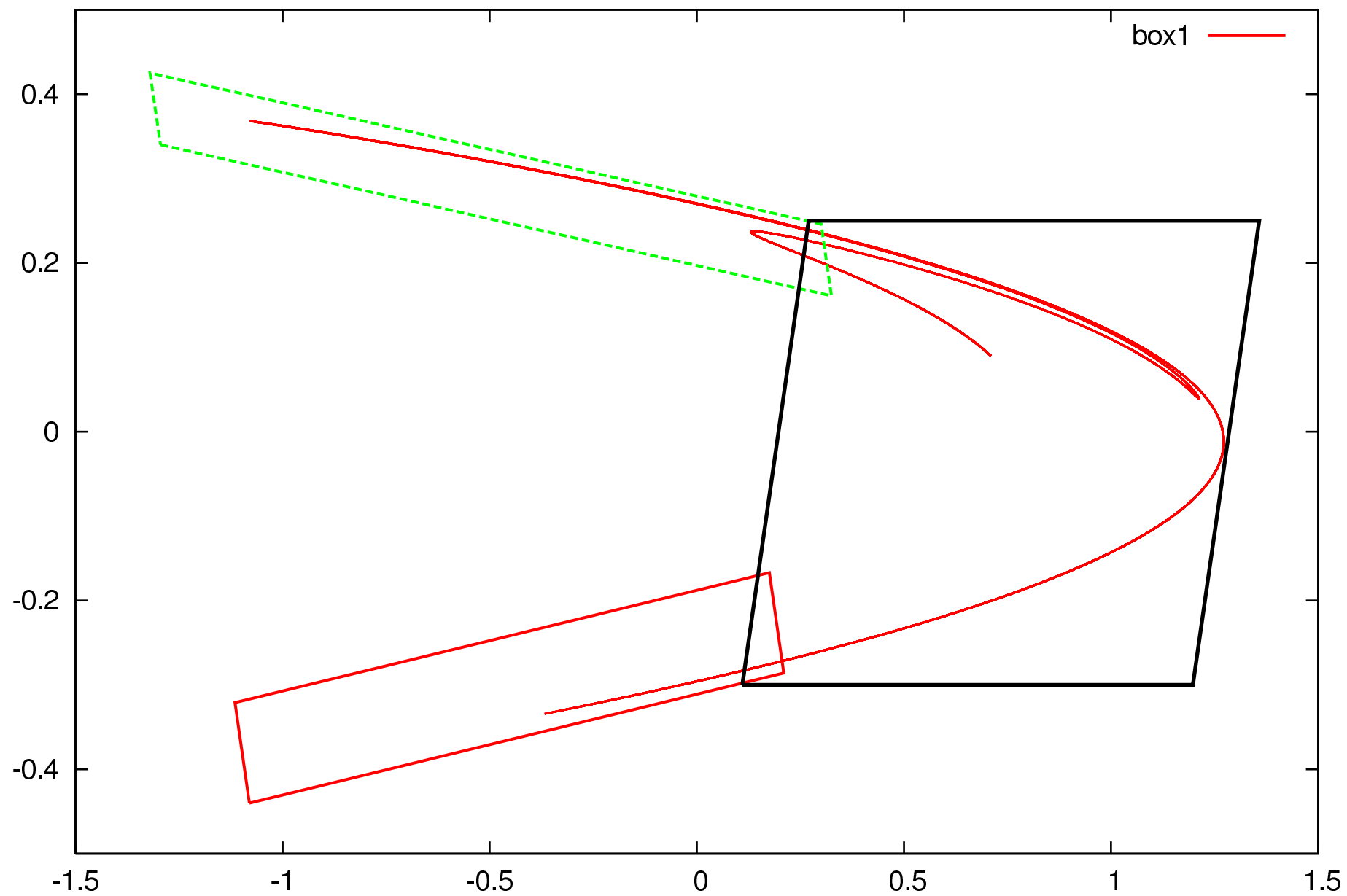


rhenon. step 5. 3/3/08

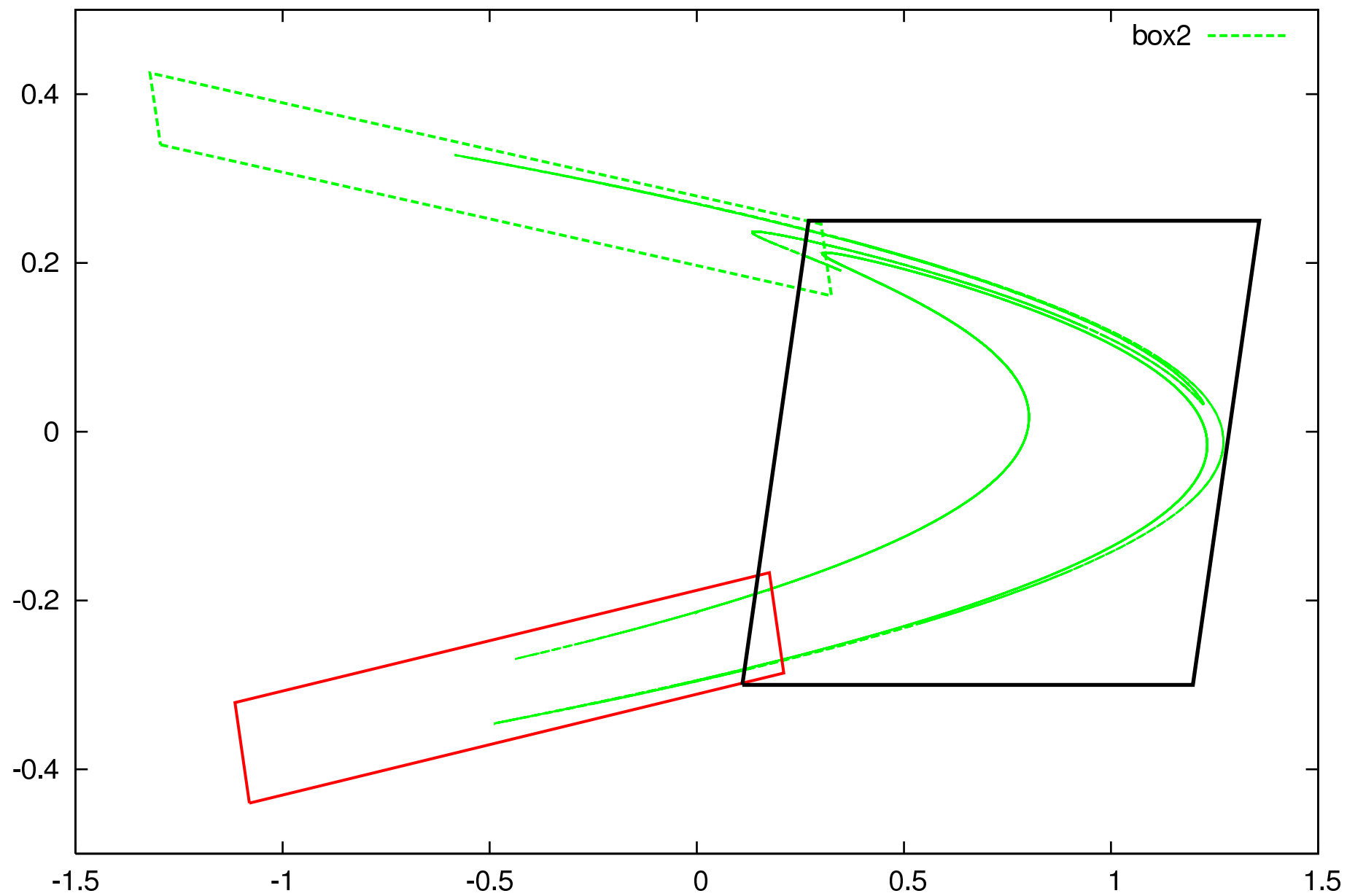




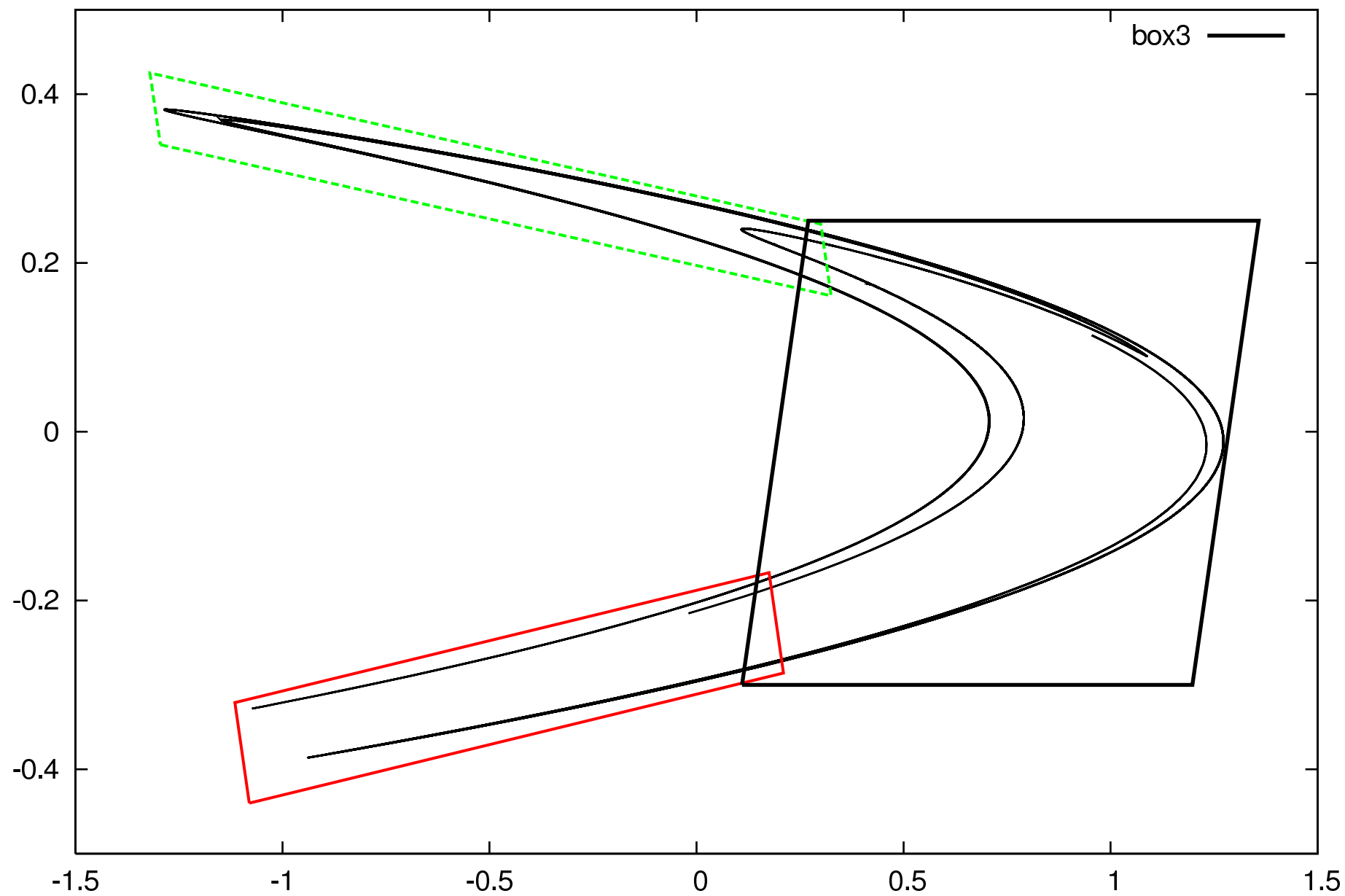
rhonon. step 5. box1. 3/3/08



rhonon. step 5. box2. 3/3/08



rhonon. step 5. box3. 3/3/08



## henon: Number of Objects

To carry out multiple mappings of the Henon map, Taylor model objects underwent the domain decomposition.

Number of Taylor model objects used for multiple mappings:

	$n$	$w$	for 5 steps	for 7 steps
box1	33	17	3	1386
box2	21	11	148	1691
box3	33	17	8	2839

Coming very soon...

Dynamic Domain Decomposition for the ODE integrator