An algorithm to reduce the number of dummy variables in affine arithmetic

Masahide Kashiwagi
kashi@waseda.jp

Waseda University, Japan

SCAN' 2012, Nobosibirsk, Russia (Sep. 28, 2012)

Outline of our presentation

1. Affine Arithmetic (AA)
2. Consider to reduce the number of εs
3. Consider to ‘intervalize’ some unimportant εs
4. How to select ‘unimportant’ εs?
5. ‘Penalty Function’ based on Hausdorff distance
6. Numerical Examples

Outline of our presentation

1. Affine Arithmetic (AA)
2. Consider to reduce the number of εs
3. Consider to 'intervalize' some unimportant εs
4. How to select 'unimportant' εs?
5. 'Penalty Function' based on Hausdorff distance
6. Numerical Examples

Outline of our presentation

1. Affine Arithmetic (AA)
2. Consider to reduce the number of εs
3. Consider to 'intervalize' some unimportant εs
4. How to select 'unimportant' εs?
5. 'Penalty Function' based on Hausdorff distance
6. Numerical Examples

Outline of our presentation

1. Affine Arithmetic (AA)
2. Consider to reduce the number of εs
3. Consider to 'intervalize' some unimportant εs
4. How to select 'unimportant' εs?
5. 'Penalty Function' based on Hausdorff distance
6. Numerical Examples

Outline of our presentation

1. Affine Arithmetic (AA)
2. Consider to reduce the number of εs
3. Consider to 'intervalize' some unimportant εs
4. How to select 'unimportant' εs?
5. 'Penalty Function' based on Hausdorff distance
6. Numerical Examples

Outline of our presentation

1. Affine Arithmetic (AA)
2. Consider to reduce the number of εs
3. Consider to 'intervalize' some unimportant εs
4. How to select 'unimportant' εs?
5. 'Penalty Function' based on Hausdorff distance
6. Numerical Examples

Affine Arithmetic (AA)

- Affine arithmetic (AA) is an extension of interval arithmetic.
- In AA, quantities are represented by affine forms:

$$a_0 + a_1\varepsilon_1 + a_2\varepsilon_2 + \cdots + a_n\varepsilon_n$$

where ε_i are dummy variables which satisfy $-1 \le \varepsilon_i \le 1$.

- In AA, number of ε gradually increases and that makes calculation slower.

Affine Arithmetic (AA)

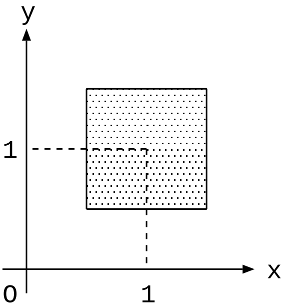
- Affine arithmetic (AA) is an extension of interval arithmetic.
- In AA, quantities are represented by affine forms:

$$a_0 + a_1\varepsilon_1 + a_2\varepsilon_2 + \cdots + a_n\varepsilon_n$$

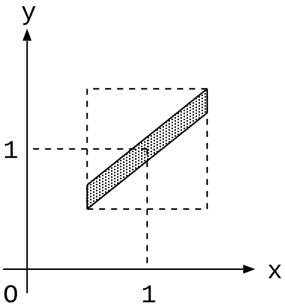
  where ε_i are dummy variables which satisfy $-1 \leq \varepsilon_i \leq 1$.
- In AA, number of ε gradually increases and that makes calculation slower.

εs represent correlation between different quantities

$$\begin{aligned} x &= 1 + 0.5\varepsilon_1 \\ y &= 1 + 0.5\varepsilon_2 \end{aligned}$$



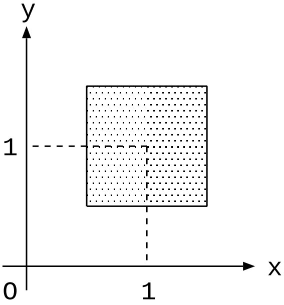
$$\begin{aligned} x &= 1 + 0.5\varepsilon_1 \\ y &= 1 + 0.4\varepsilon_1 + 0.1\varepsilon_2 \end{aligned}$$



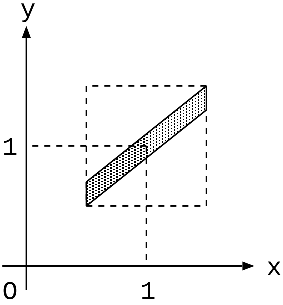
Same interval, different joint range.

εs represent correlation between different quantities

$$
\begin{aligned}
x &= 1 + 0.5\varepsilon_1 \\
y &= 1 + 0.5\varepsilon_2
\end{aligned}
$$



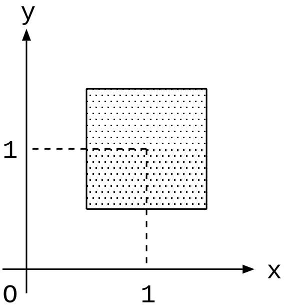
$$
\begin{aligned}
x &= 1 + 0.5\varepsilon_1 \\
y &= 1 + 0.4\varepsilon_1 + 0.1\varepsilon_2
\end{aligned}
$$



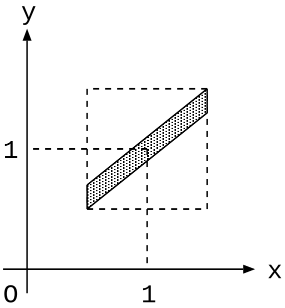
Same interval, different joint range.

εs represent correlation between different quantities

$$
\begin{aligned}
x &= 1 + 0.5\varepsilon_1 \\
y &= 1 + 0.5\varepsilon_2
\end{aligned}
$$



$$
\begin{aligned}
x &= 1 + 0.5\varepsilon_1 \\
y &= 1 + 0.4\varepsilon_1 + 0.1\varepsilon_2
\end{aligned}
$$

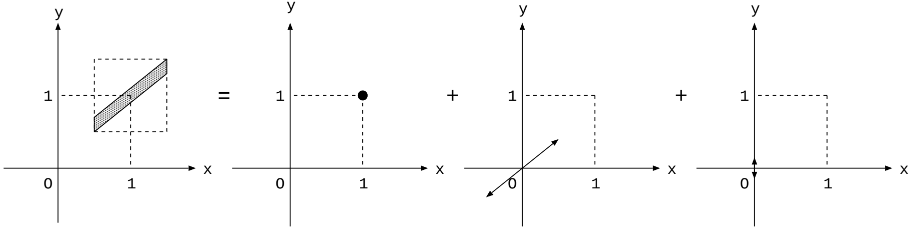


Same interval, different joint range.

Minkowski sum

We can recognize the range as the Minkowski sum of center point and line segments.

$$
\begin{aligned}
x &= 1 \quad +0.5\varepsilon_1 \\
y &= 1 \quad +0.4\varepsilon_1 \quad +0.1\varepsilon_2
\end{aligned}
$$



return

Convert between interval

interval → affine

$$\begin{pmatrix} [\underline{x_1}, \overline{x_1}] \\ [\underline{x_2}, \overline{x_2}] \\ \vdots \\ [\underline{x_n}, \overline{x_n}] \end{pmatrix} \Longrightarrow \begin{pmatrix} \frac{\overline{x_1}+\underline{x_1}}{2} + \frac{\overline{x_1}-\underline{x_1}}{2}\varepsilon_1 \\ \frac{\overline{x_2}+\underline{x_2}}{2} + \frac{\overline{x_2}-\underline{x_2}}{2}\varepsilon_2 \\ \vdots \\ \frac{\overline{x_n}+\underline{x_n}}{2} + \frac{\overline{x_n}-\underline{x_n}}{2}\varepsilon_n \end{pmatrix}$$

affine → interval

$$x = a_0 + a_1\varepsilon_1 + \cdots + a_n\varepsilon_n$$

$$\Downarrow$$

$$[a_0 - \delta, a_0 + \delta] \qquad , \quad (\delta = \sum_{i=1}^{n} |a_i|)$$

Linear Operation is easy

$$
\begin{aligned}
x &= x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \\
y &= y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n
\end{aligned}
$$

Addition, Subtraction

$$
\begin{aligned}
x \pm y &= (x_0 \pm y_0) + (x_1 \pm y_1)\varepsilon_1 + \cdots + (x_n \pm y_n)\varepsilon_n \\
x \pm \alpha &= (x_0 \pm \alpha) + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \quad .
\end{aligned}
$$

Constant Multiplication

$$
\alpha x = (\alpha x_0) + (\alpha x_1)\varepsilon_1 + \cdots + (\alpha x_n)\varepsilon_n \quad .
$$

Nonlinear Unary Operations (Standard Functions)

f: Nonlinear Unary Operation such as exp, log, $\cdots$.
Consider to calculate $z = f(x)$ for affine variable

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$$

1. Calculate interval I (range of x) as

$$I = [x_0 - \delta, x_0 + \delta], \quad \delta = \sum_{i=1}^{n} |x_i| \quad ,$$

2. Obtain $ax + b$ (a linear approximation of f on I) and maximum error

$$\delta = \max_{x \in I} |f(x) - (ax + b)|$$

3. Obtain the result z as

$$a(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) + b + \delta\varepsilon_{n+1}$$

Nonlinear Unary Operations (Standard Functions)

f: Nonlinear Unary Operation such as $\exp, \log, \cdots$.
Consider to calculate $z = f(x)$ for affine variable

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$$

1. Calculate interval I (range of x) as

$$I = [x_0 - \delta, x_0 + \delta], \quad \delta = \sum_{i=1}^{n} |x_i| \quad ,$$

2. Obtain $ax + b$ (a linear approximation of f on I) and maximum error

$$\delta = \max_{x \in I} |f(x) - (ax + b)|$$

3. Obtain the result z as

$$a(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) + b + \delta\varepsilon_{n+1}$$

Nonlinear Unary Operations (Standard Functions)

f: Nonlinear Unary Operation such as $\exp, \log, \cdots$.
Consider to calculate $z = f(x)$ for affine variable

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$$

1. Calculate interval I (range of x) as

$$I = [x_0 - \delta, x_0 + \delta], \quad \delta = \sum_{i=1}^{n} |x_i| \quad ,$$

2. Obtain $ax + b$ (a linear approximation of f on I) and maximum error

$$\delta = \max_{x \in I} |f(x) - (ax + b)|$$

3. Obtain the result z as

$$a(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) + b + \delta\varepsilon_{n+1}$$

Nonlinear Unary Operations (Standard Functions)

f: Nonlinear Unary Operation such as $\exp, \log, \cdots$.
Consider to calculate $z = f(x)$ for affine variable

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$$

1. Calculate interval I (range of x) as

$$I = [x_0 - \delta, x_0 + \delta], \quad \delta = \sum_{i=1}^{n} |x_i| \quad ,$$

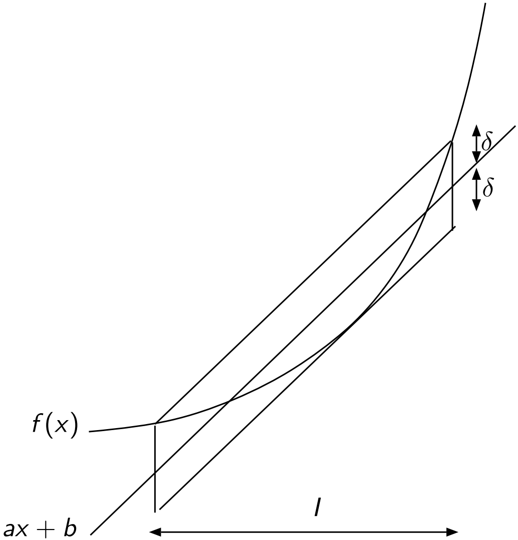
2. Obtain $ax + b$ (a linear approximation of f on I) and maximum error

$$\delta = \max_{x \in I} |f(x) - (ax + b)|$$

3. Obtain the result z as

$$a(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) + b + \delta\varepsilon_{n+1}$$

linear approximation $ax + b$ and error δ



Nonlinear Binary Operations

Consider linear approximation $ax + by + c$ for binary operator $g(x, y)$.
(almost same as the case of unary operations)

Multiplication

$$
\begin{aligned}
z &= y_0 x + x_0 y - x_0 y_0 + \delta_x \delta_y \varepsilon_{n+1} \\
&= x_0 y_0 + \sum_{i=1}^{n} (y_0 x_i + x_0 y_i) \varepsilon_i \\
&\quad + (\sum_{i=1}^{n} |x_i|)(\sum_{i=1}^{n} |y_i|) \varepsilon_{n+1}
\end{aligned}
$$

Example for comparison of inclusion tightness: Henon map

Henon Map: a discrete-time dynamical system:

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} 1 - ax_i^2 + y_i \\ bx_i \end{pmatrix}$$

- It is known that Henon map is chaotic at $b = 0.3$, $a \geq 1.06$.
- We use parameter $b = 0.3$, $a = 1.05$ (near chaotic).

Example for comparison of inclusion tightness: Henon map

Henon Map: a discrete-time dynamical system:

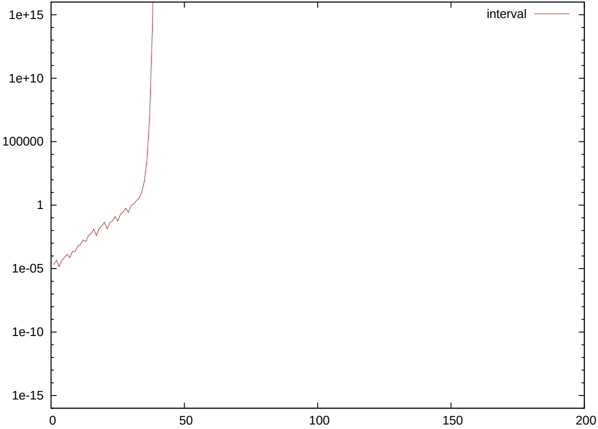
$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} 1 - ax_i^2 + y_i \\ bx_i \end{pmatrix}$$

- It is known that Henon map is chaotic at $b = 0.3$, $a \geq 1.06$.
- We use parameter $b = 0.3$, $a = 1.05$ (near chaotic).

Calculating Henon map by interval and affine(1)

- horizontal axis: number of iteration
- vertival axis: $\max(\mathrm{width}(x), \mathrm{width}(y))$
- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$

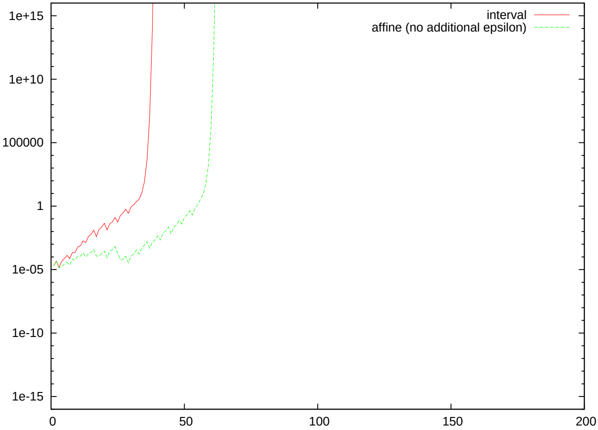
return



Calculating Henon map by interval and affine(1)

- horizontal axis: number of iteration
- vertival axis: $\max(\mathrm{width}(x), \mathrm{width}(y))$
- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$

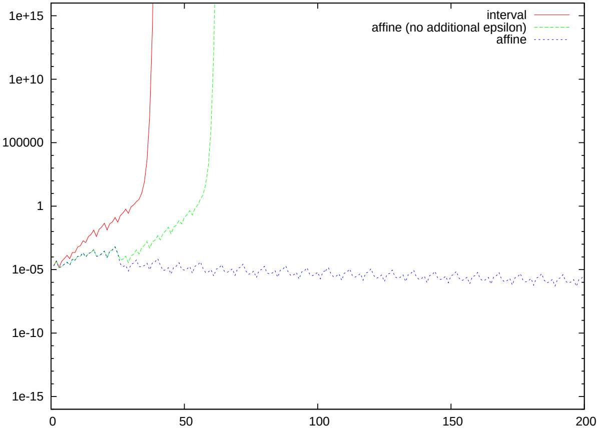
return



Calculating Henon map by interval and affine(1)

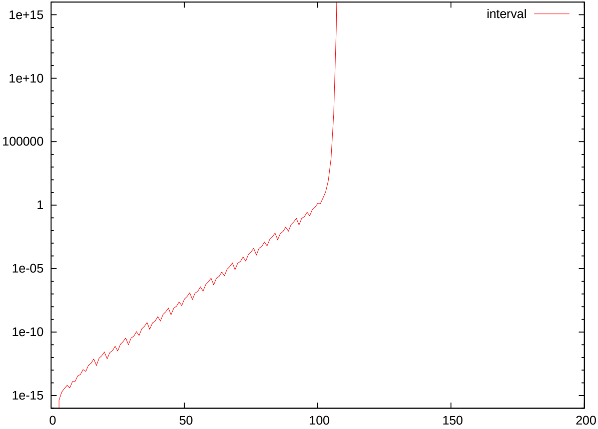
- horizontal axis: number of iteration
- vertival axis: $\max(\mathrm{width}(x), \mathrm{width}(y))$
- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$

return



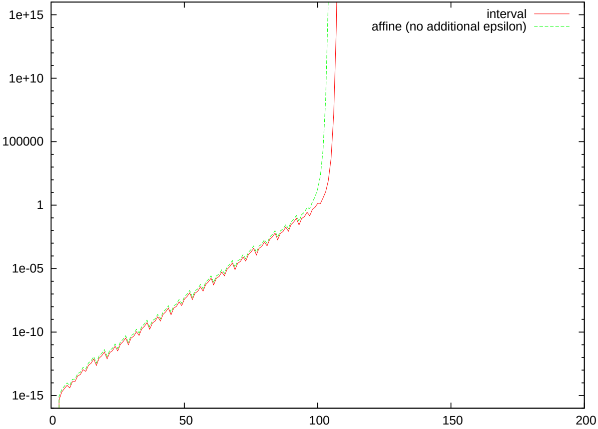
Calculating Henon map by interval and affine(2)

- horizontal axis: number of iteration
- vertival axis: $\max(\mathrm{width}(x), \mathrm{width}(y))$
- initial value: $(x(0), y(0)) = ([0, 0], [0, 0])$



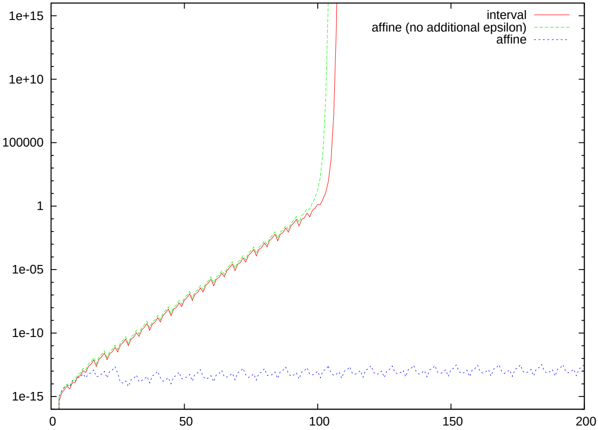
Calculating Henon map by interval and affine(2)

- horizontal axis: number of iteration
- vertival axis: $\max(\mathrm{width}(x), \mathrm{width}(y))$
- initial value: $(x(0), y(0)) = ([0, 0], [0, 0])$



Calculating Henon map by interval and affine(2)

- horizontal axis: number of iteration
- vertival axis: $\max(\mathrm{width}(x), \mathrm{width}(y))$
- initial value: $(x(0), y(0)) = ([0, 0], [0, 0])$



Calculating Henon map by interval and affine(3)

	calculation time (msec)	maximum number of εs
interval	0.016	0
affine (no additional ε)	0.111	2
affine	0.601	202

- CPU: core i7 2640M (2.8GHz)
- OS: ubuntu 10.04 LTS (64bit)
- software: GNU C++, boost.interval, boost.ublas

Problem

- Affine arithmetic has very high ability to get tight inclusion.
- We must introduce new dummy variable ε_{n+1} for each nonlinear operation.
- Number of εs gradually increases and that makes calculation slower.
- How can we reduce the number of εs without loss of tight inclusion?

Problem

- Affine arithmetic has very high ability to get tight inclusion.
- We must introduce new dummy variable ε_{n+1} for each nonlinear operation.
- Number of εs gradually increases and that makes calculation slower.
- How can we reduce the number of εs without loss of tight inclusion?

Problem

- Affine arithmetic has very high ability to get tight inclusion.
- We must introduce new dummy variable ε_{n+1} for each nonlinear operation.
- Number of εs gradually increases and that makes calculation slower.
- How can we reduce the number of εs without loss of tight inclusion?

Problem

- Affine arithmetic has very high ability to get tight inclusion.
- We must introduce new dummy variable ε_{n+1} for each nonlinear operation.
- Number of εs gradually increases and that makes calculation slower.
- How can we reduce the number of εs without loss of tight inclusion?

Policy of Epsilon-Reduction(1)

Consider p affine variables which have q dummy εs:

$$a_{10} + a_{11}\varepsilon_1 + \cdots + a_{1q}\varepsilon_q$$

$$\vdots$$

$$a_{p0} + a_{p1}\varepsilon_1 + \cdots + a_{pq}\varepsilon_q$$

we can reduce the number of ε by 'intervalize' several εs. Let S be a index set of εs which we want to erase, we can erase εs by substituting as follows:

$$\sum_{i \in S} a_{1i}\varepsilon_i \rightarrow \left(\sum_{i \in S} |a_{1i}|\right)\varepsilon_{q+1}$$

$$\vdots$$

$$\sum_{i \in S} a_{pi}\varepsilon_i \rightarrow \left(\sum_{i \in S} |a_{pi}|\right)\varepsilon_{q+p}$$

Here, p new εs are added in order to represent the newly generated intervals.

Policy of Epsilon-Reduction(2)

In the following, we consider to reduce number of εs to r .

Overestimation should be as small as possible .

We keep $r-p$ εs which have big 'intervalize penalty',

and intervalize $q-(r-p)$ εs which have small 'intervalize penalty'.

Then we can reduce the total number of εs to $(r-p)+p=r$:

$$\overbrace{\sum_{i\notin S} a_{1i}\varepsilon_i}^{r-p} + \overbrace{(\sum_{i\in S}|a_{1i}|)\varepsilon_{q+1}}^{q-(r-p)}$$

$$\vdots$$

$$\sum_{i\notin S} a_{pi}\varepsilon_i + (\sum_{i\in S}|a_{pi}|)\varepsilon_{q+p}$$

What is intervalize penalty?

Policy of Epsilon-Reduction(2)

In the following, we consider to reduce number of εs to r .
Overestimation should be as small as possible .
We keep $r-p$ εs which have big 'intervalize penalty',
and intervalize $q-(r-p)$ εs which have small 'intervalize penalty'.
Then we can reduce the total number of εs to $(r-p)+p=r$:

$$\overbrace{\sum_{i \notin S} a_{1i}\varepsilon_i}^{r-p} + \overbrace{(\sum_{i \in S} |a_{1i}|)\varepsilon_{q+1}}^{q-(r-p)}$$

$$\vdots$$

$$\sum_{i \notin S} a_{pi}\varepsilon_i + (\sum_{i \in S} |a_{pi}|)\varepsilon_{q+p}$$

What is intervalize penalty?

Policy of Epsilon-Reduction(2)

In the following, we consider to reduce number of εs to r .
Overestimation should be as small as possible .
We keep $r - p$ εs which have big 'intervalize penalty',
and intervalize $q - (r - p)$ εs which have small 'intervalize penalty'.
Then we can reduce the total number of εs to $(r - p) + p = r$:

$$\overbrace{\sum_{i \notin S} a_{1i}\varepsilon_i}^{r-p} + \overbrace{(\sum_{i \in S} |a_{1i}|)\varepsilon_{q+1}}^{q-(r-p)}$$

$$\vdots$$

$$\sum_{i \notin S} a_{pi}\varepsilon_i + (\sum_{i \in S} |a_{pi}|)\varepsilon_{q+p}$$

What is intervalize penalty?

Policy of Epsilon-Reduction(2)

In the following, we consider to reduce number of εs to r .

Overestimation should be as small as possible .

We keep $r-p$ εs which have big 'intervalize penalty',

and intervalize $q-(r-p)$ εs which have small 'intervalize penalty'.

Then we can reduce the total number of εs to $(r-p)+p=r$:

$$\overbrace{\sum_{i \notin S} a_{1i}\varepsilon_i}^{r-p} + \overbrace{(\sum_{i \in S} |a_{1i}|)\varepsilon_{q+1}}^{q-(r-p)}$$

$$\vdots$$

$$\sum_{i \notin S} a_{pi}\varepsilon_i + (\sum_{i \in S} |a_{pi}|)\varepsilon_{q+p}$$

What is intervalize penalty?

Policy of Epsilon-Reduction(2)

In the following, we consider to reduce number of εs to r .
Overestimation should be as small as possible .
We keep $r-p$ εs which have big 'intervalize penalty',
and intervalize $q-(r-p)$ εs which have small 'intervalize penalty'.
Then we can reduce the total number of εs to $(r-p)+p=r$:

$$\overbrace{\sum_{i\notin S} a_{1i}\varepsilon_i}^{r-p} + \overbrace{(\sum_{i\in S}|a_{1i}|)\varepsilon_{q+1}}^{q-(r-p)}$$

$$\vdots$$

$$\sum_{i\notin S} a_{pi}\varepsilon_i + (\sum_{i\in S}|a_{pi}|)\varepsilon_{q+p}$$

What is intervalize penalty?

Policy of Epsilon-Reduction(2)

In the following, we consider to reduce number of εs to r .
Overestimation should be as small as possible .
We keep $r - p$ εs which have big 'intervalize penalty',
and intervalize $q - (r - p)$ εs which have small 'intervalize penalty'.
Then we can reduce the total number of εs to $(r - p) + p = r$:

$$\overbrace{\sum_{i \notin S} a_{1i}\varepsilon_i}^{r-p} + \overbrace{(\sum_{i \in S} |a_{1i}|)\varepsilon_{q+1}}^{q-(r-p)}$$

$$\vdots$$

$$\sum_{i \notin S} a_{pi}\varepsilon_i + (\sum_{i \in S} |a_{pi}|)\varepsilon_{q+p}$$

What is intervalize penalty?

Example of Epsilon Reduction

We have

$$
\begin{array}{rcl}
x &=& 1 \quad +\varepsilon_1 \quad -\varepsilon_2 \quad +0.1\varepsilon_3 \quad -0.3\varepsilon_4 \quad\quad\quad\quad +0.1\varepsilon_6 \quad +0.5\varepsilon_7 \\
y &=& 1 \quad +0.2\varepsilon_1 \quad +\varepsilon_2 \quad +0.05\varepsilon_3 \quad -0.3\varepsilon_4 \quad +0.5\varepsilon_5 \quad +0.03\varepsilon_6 \quad -0.2\varepsilon_7
\end{array}
$$

and let index set $S = \{3, 5, 6, 7\}$ then we can erase $\varepsilon_3, \varepsilon_5, \varepsilon_6, \varepsilon_7$ as

$$
\begin{aligned}
\begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + (|0.1| + |0| + |0.1| + |0.5|)\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + (|0.05| + |0.5| + |0.03| + |0.2|)\varepsilon_9 \end{pmatrix} \\
&= \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + 0.7\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + 0.78\varepsilon_9 \end{pmatrix}
\end{aligned}
$$

7 εs $\to$ keep 3 εs and intervalize 4 εs $\to$ 5 εs.

Which ε should be intervalized / kept to minimize overestimation?

Example of Epsilon Reduction

We have

$$\begin{array}{rclcccccccc} x &=& 1 & +\varepsilon_1 & -\varepsilon_2 & +0.1\varepsilon_3 & -0.3\varepsilon_4 & & +0.1\varepsilon_6 & +0.5\varepsilon_7 \\ y &=& 1 & +0.2\varepsilon_1 & +\varepsilon_2 & +0.05\varepsilon_3 & -0.3\varepsilon_4 & +0.5\varepsilon_5 & +0.03\varepsilon_6 & -0.2\varepsilon_7 \end{array}$$

and let index set $S = \{3, 5, 6, 7\}$ then we can erase $\varepsilon_3, \varepsilon_5, \varepsilon_6, \varepsilon_7$ as

$$\begin{aligned} \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + (|0.1| + |0| + |0.1| + |0.5|)\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + (|0.05| + |0.5| + |0.03| + |0.2|)\varepsilon_9 \end{pmatrix} \\ &= \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + 0.7\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + 0.78\varepsilon_9 \end{pmatrix} \end{aligned}$$

7 εs $\to$ keep 3 εs and intervalize 4 εs $\to$ 5 εs.
Which ε should be intervalized / kept to minimize overestimation?

Example of Epsilon Reduction

We have

$$\begin{array}{lclcccccccc} x & = & 1 & +\varepsilon_1 & -\varepsilon_2 & +0.1\varepsilon_3 & -0.3\varepsilon_4 & & +0.1\varepsilon_6 & +0.5\varepsilon_7 \\ y & = & 1 & +0.2\varepsilon_1 & +\varepsilon_2 & +0.05\varepsilon_3 & -0.3\varepsilon_4 & +0.5\varepsilon_5 & +0.03\varepsilon_6 & -0.2\varepsilon_7 \end{array}$$

and let index set $S = \{3, 5, 6, 7\}$ then we can erase $\varepsilon_3, \varepsilon_5, \varepsilon_6, \varepsilon_7$ as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + (|0.1| + |0| + |0.1| + |0.5|)\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + (|0.05| + |0.5| + |0.03| + |0.2|)\varepsilon_9 \end{pmatrix}$$

$$= \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + 0.7\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + 0.78\varepsilon_9 \end{pmatrix}$$

7 εs $\to$ keep 3 εs and intervalize 4 εs $\to$ 5 εs.

Which ε should be intervalized / kept to minimize overestimation?

Example of Epsilon Reduction

We have

$$\begin{array}{lclcccccccc} x & = & 1 & +\varepsilon_1 & -\varepsilon_2 & +0.1\varepsilon_3 & -0.3\varepsilon_4 & & +0.1\varepsilon_6 & +0.5\varepsilon_7 \\ y & = & 1 & +0.2\varepsilon_1 & +\varepsilon_2 & +0.05\varepsilon_3 & -0.3\varepsilon_4 & +0.5\varepsilon_5 & +0.03\varepsilon_6 & -0.2\varepsilon_7 \end{array}$$

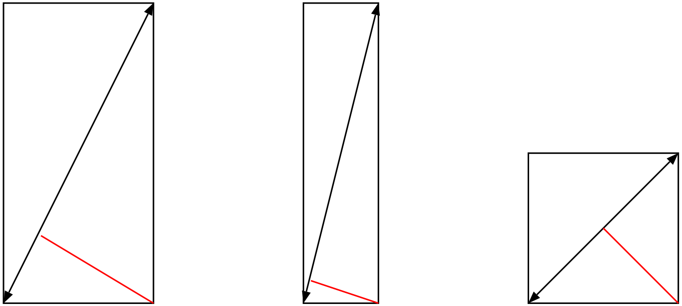
and let index set $S = \{3, 5, 6, 7\}$ then we can erase $\varepsilon_3, \varepsilon_5, \varepsilon_6, \varepsilon_7$ as

$$\begin{aligned} \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + (|0.1| + |0| + |0.1| + |0.5|)\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + (|0.05| + |0.5| + |0.03| + |0.2|)\varepsilon_9 \end{pmatrix} \\ &= \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + 0.7\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + 0.78\varepsilon_9 \end{pmatrix} \end{aligned}$$

7 εs $\to$ keep 3 εs and intervalize 4 εs $\to$ 5 εs.

Which ε should be intervalized / kept to minimize overestimation?

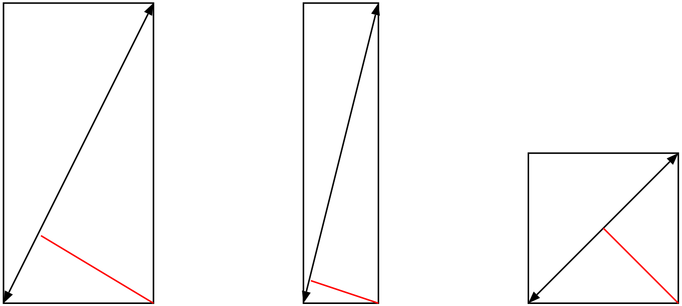
Intervalize Penalty?



If we intervalize some ε then the line segment made by the ε is covered by the hyper-rectangular. ▸ Minkowski

We can regard the Hausdorff distance between the line segment and hyper-rectangular as the intervalize penalty of the ε.

Intervalize Penalty?



If we intervalize some ε then the line segment made by the ε is covered by the hyper-rectangular. ▸ Minkowski

We can regard the Hausdorff distance between the line segment and hyper-rectangular as the intervalize penalty of the ε.

Penalty Function

Let vectors $v_0, \cdots, v_q \in \mathbf{R}^p$ be $v_i = (a_{i1}, \cdots, a_{ip})^T$

$$a_{10} + a_{11}\varepsilon_1 + \cdots + a_{1q}\varepsilon_q$$

$$\vdots$$

$$a_{p0} + a_{p1}\varepsilon_1 + \cdots + a_{pq}\varepsilon_q$$

$$\Downarrow$$

$$v_0 + v_1\varepsilon_1 + \cdots + v_q\varepsilon_q$$

Penalty Function

For vector $v = (a_1, \cdots, a_p)^T$ we define penalty function P as follows:

- When $a_1 = a_2 = \cdots = a_p = 0$, we define $P(v) = 0$
- Otherwise, let a_s, a_t be the first and second values in the order of absolute values $|a_i|$. That is, $|a_s| \geq |a_t| \geq |a_i| \quad (i \neq s, t)$ hold. Then we define $P(v) = \dfrac{|a_s| \cdot |a_t|}{|a_s| + |a_t|}$

Penalty function and Hausdorff distance

Let $v = (a_1, \cdots, a_p)^T \in \mathbf{R}^p$ and norm of $\mathbf{R}^p$ be maximum norm. Let $L \subset \mathbf{R}^p$ be a line segment defined by

$$(a_1, \cdots, a_p)^T \varepsilon \quad (-1 \le \varepsilon \le 1)$$

and let $B \subset \mathbf{R}^p$ be a hyper-rectangular defined by

$$(a_1\varepsilon_1, \cdots, a_p\varepsilon_p)^T \quad (-1 \le \varepsilon_i \le 1)$$

Then Hausdorff distance between L and B becomes

$$H(L, B) = 2P(v)$$

Hausdorff distance

$$H(X, Y) = \max\{\sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y)\}$$

Example of Penalty Function(1)

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix}$$

$$2, 0.5, 3, 1 \quad \text{(absolute value)}$$

$$3 \geq 2 \geq 1 \geq 0.5 \quad \text{(sort)}$$

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix} = \frac{3 \times 2}{3 + 2} = 6/5 = 1.2$$

Example of Penalty Function(1)

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix}$$

$$2, 0.5, 3, 1 \quad \text{(absolute value)}$$

$$3 \geq 2 \geq 1 \geq 0.5 \quad \text{(sort)}$$

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix} = \frac{3 \times 2}{3+2} = 6/5 = 1.2$$

Example of Penalty Function(1)

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix}$$

$$2, 0.5, 3, 1 \quad \text{(absolute value)}$$

$$3 \geq 2 \geq 1 \geq 0.5 \quad \text{(sort)}$$

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix} = \frac{3 \times 2}{3 + 2} = 6/5 = 1.2$$

Example of Penalty Function(1)

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix}$$

$$2, 0.5, 3, 1 \quad \text{(absolute value)}$$

$$3 \geq 2 \geq 1 \geq 0.5 \quad \text{(first and second value)}$$

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix} = \frac{3 \times 2}{3 + 2} = 6/5 = 1.2$$

Example of Penalty Function(1)

$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix}$$

$$2, 0.5, 3, 1 \quad \text{(absolute value)}$$

$$3 \geq 2 \geq 1 \geq 0.5 \quad \text{(first and second value)}$$

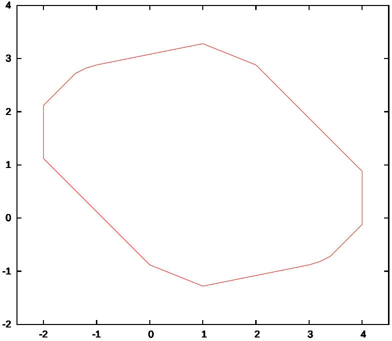
$$P\begin{pmatrix} 2 \\ -0.5 \\ -3 \\ 1 \end{pmatrix} = \frac{3 \times 2}{3 + 2} = 6/5 = 1.2$$

Example of Penalty Function(2)

$$
\begin{array}{ccccccccccc}
x & = & 1 & +\varepsilon_1 & -\varepsilon_2 & +0.1\varepsilon_3 & -0.3\varepsilon_4 & & +0.1\varepsilon_6 & +0.5\varepsilon_7 \\
y & = & 1 & +0.2\varepsilon_1 & +\varepsilon_2 & +0.05\varepsilon_3 & -0.3\varepsilon_4 & +0.5\varepsilon_5 & +0.03\varepsilon_6 & -0.2\varepsilon_7
\end{array}
$$

$$
\begin{aligned}
P(v_1) &= \frac{1 \times 0.2}{1 + 0.2} = 1/6 = 0.1666\cdots \\
P(v_2) &= \frac{1 \times 1}{1 + 1} = 1/2 = 0.5 \\
P(v_3) &= \frac{0.1 \times 0.05}{0.1 + 0.05} = 1/30 = 0.0333\cdots \\
P(v_4) &= \frac{0.3 \times 0.3}{0.3 + 0.3} = 3/20 = 0.15 \\
P(v_5) &= \frac{0 \times 0.5}{0 + 0.5} = 0 \\
P(v_6) &= \frac{0.1 \times 0.03}{0.1 + 0.03} = 3/130 = 0.0230\cdots \\
P(v_7) &= \frac{0.5 \times 0.2}{0.5 + 0.2} = 1/7 = 0.142\cdots
\end{aligned}
$$

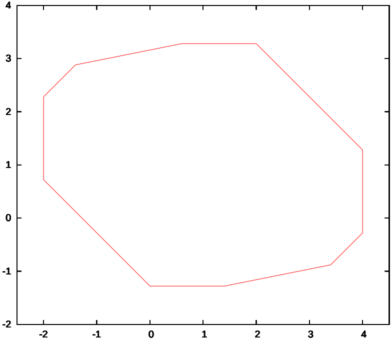
Simple Example



(original: number of εs = 7)

$$
\begin{array}{ccccccccccc}
x & = & 1 & +\varepsilon_1 & -\varepsilon_2 & +0.1\varepsilon_3 & -0.3\varepsilon_4 & & +0.1\varepsilon_6 & +0.5\varepsilon_7 \\
y & = & 1 & +0.2\varepsilon_1 & +\varepsilon_2 & +0.05\varepsilon_3 & -0.3\varepsilon_4 & +0.5\varepsilon_5 & +0.03\varepsilon_6 & -0.2\varepsilon_7
\end{array}
$$

Simple Example

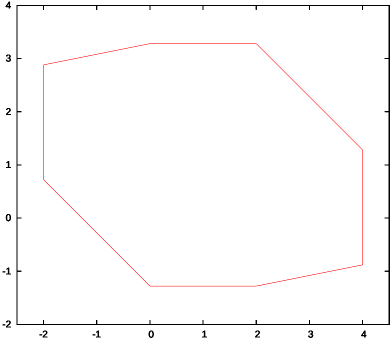


(number of εs is reduced to 5)

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + (|0.1| + |0| + |0.1| + |0.5|)\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + (|0.05| + |0.5| + |0.03| + |0.2|)\varepsilon_9 \end{pmatrix}$$

$$= \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 - 0.3\varepsilon_4 + 0.7\varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 - 0.3\varepsilon_4 + 0.78\varepsilon_9 \end{pmatrix}$$

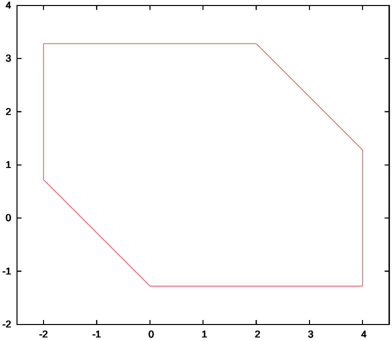
Simple Example



(number of εs is reduced to 4)

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + 1\varepsilon_1 - \varepsilon_2 + \varepsilon_8 \\ 1 + 0.2\varepsilon_1 + \varepsilon_2 + 1.08\varepsilon_9 \end{pmatrix}$$

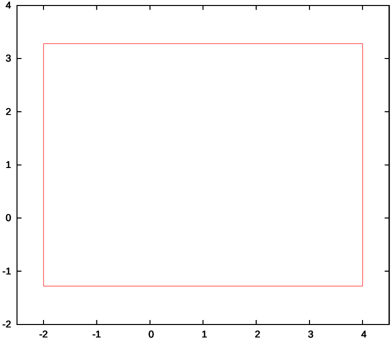
Simple Example



(number of εs is reduced to 3)

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 - \varepsilon_2 + 2\varepsilon_8 \\ 1 + \varepsilon_2 + 1.28\varepsilon_9 \end{pmatrix}$$

Simple Example

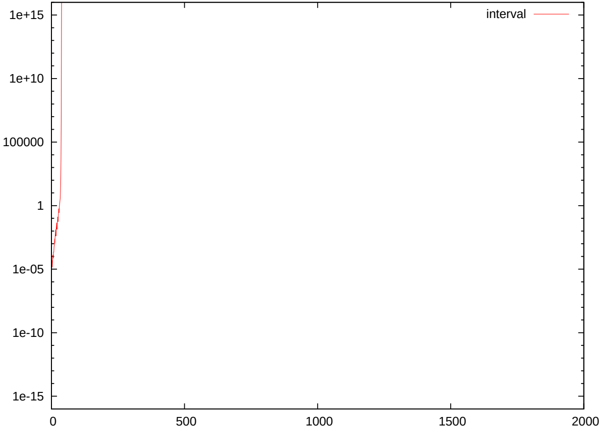


(number of εs is reduced to 2)

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + 3\varepsilon_8 \\ 1 + 2.28\varepsilon_9 \end{pmatrix}$$

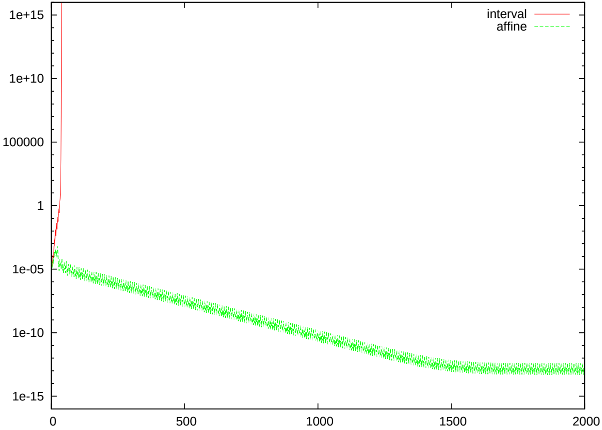
Calculating Henon map with epsilon reduction(1)

- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$ ▸ Henon
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



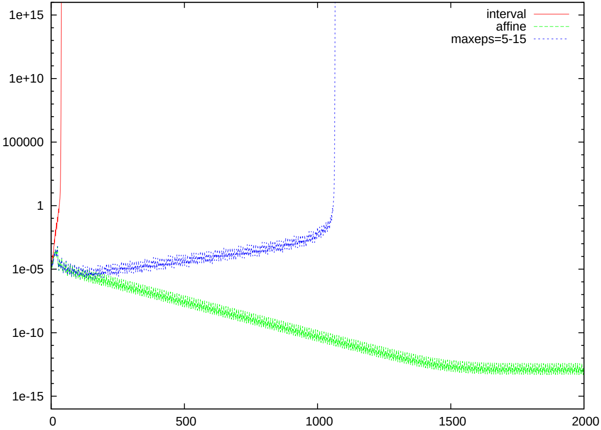
Calculating Henon map with epsilon reduction(1)

- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$ ▸ Henon
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



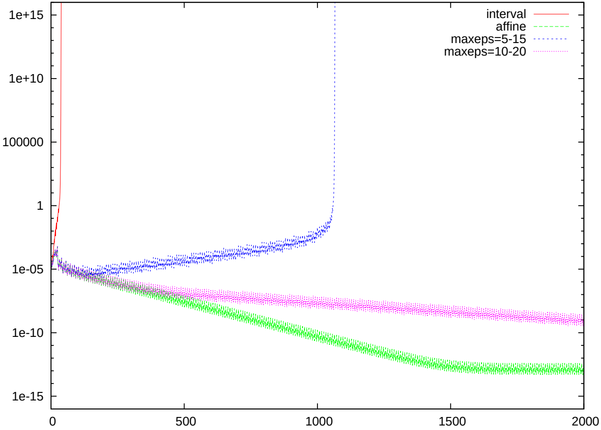
Calculating Henon map with epsilon reduction(1)

- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$ Henon
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



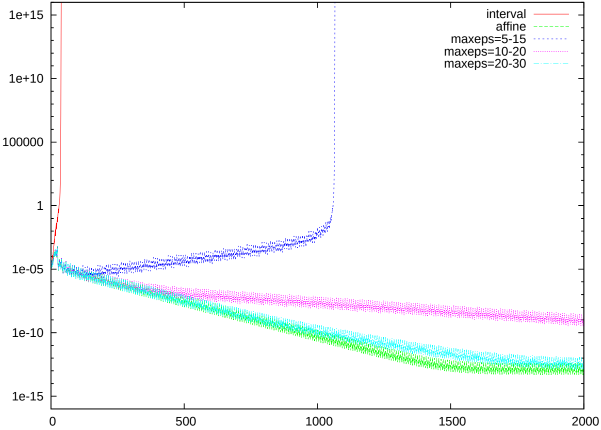
Calculating Henon map with epsilon reduction(1)

- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$ Henon
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



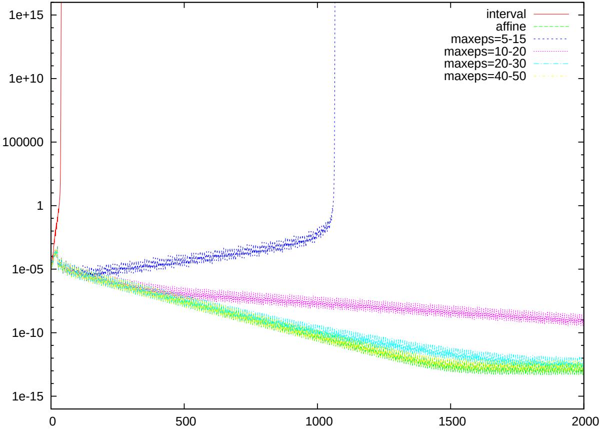
Calculating Henon map with epsilon reduction(1)

- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$ ▸ Henon
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



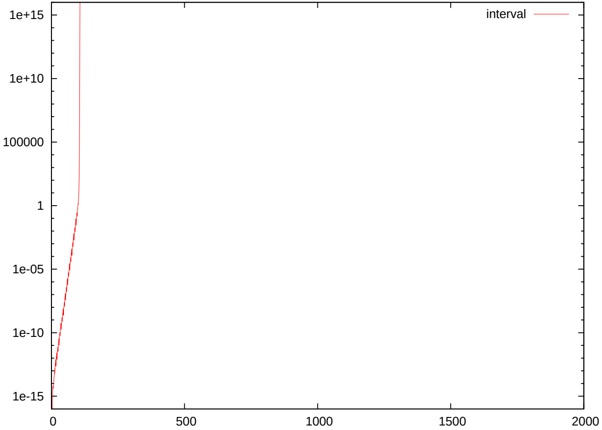
Calculating Henon map with epsilon reduction(1)

- initial value: $(x(0), y(0)) = ([-10^{-5}, 10^{-5}], [-10^{-5}, 10^{-5}])$ Henon
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



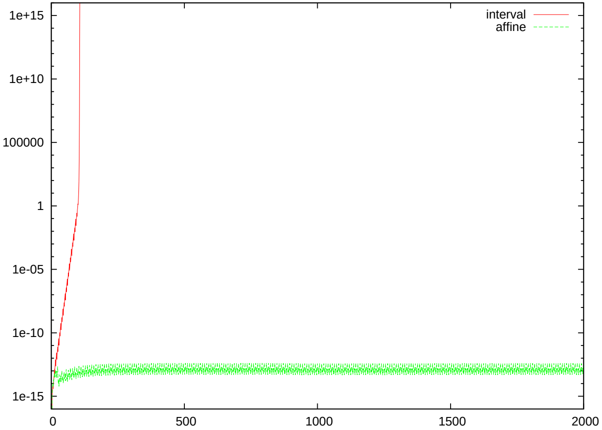
Calculating Henon map with epsilon reduction(2)

- initial value: $(x(0), y(0)) = ([0,0],[0,0])$
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



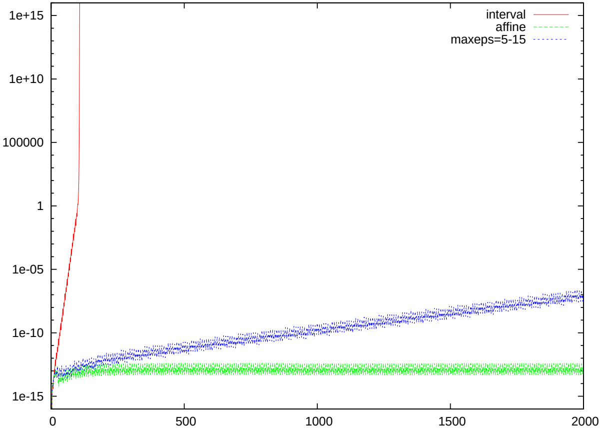
Calculating Henon map with epsilon reduction(2)

- initial value: $(x(0), y(0)) = ([0,0],[0,0])$
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



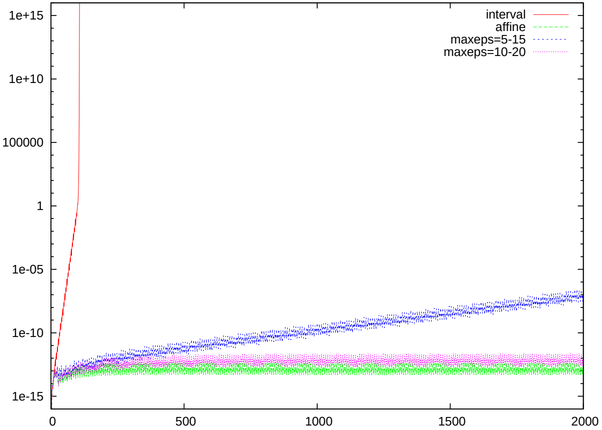
Calculating Henon map with epsilon reduction(2)

- initial value: $(x(0), y(0)) = ([0,0],[0,0])$
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



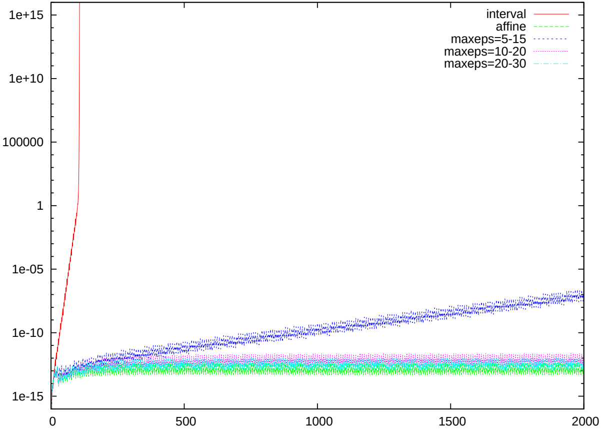
Calculating Henon map with epsilon reduction(2)

- initial value: $(x(0), y(0)) = ([0,0],[0,0])$
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



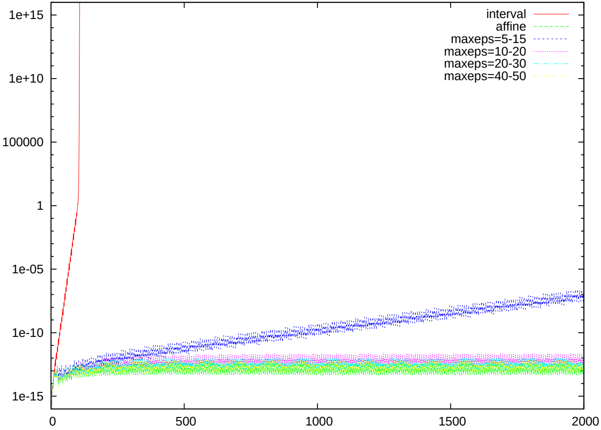
Calculating Henon map with epsilon reduction(2)

- initial value: $(x(0), y(0)) = ([0, 0], [0, 0])$
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



Calculating Henon map with epsilon reduction(2)

- initial value: $(x(0), y(0)) = ([0, 0], [0, 0])$
- 'maxeps = n-m' means that if number of $\varepsilon \geq m$ then reduce the number of ε to n.



Calculating Henon map with epsilon reduction(3)

	calculation time (msec)	maximum number of εs
interval	0.15	0
affine	35.77	2002
maxeps=5-15	2.1	15
maxeps=10-20	2.81	20
maxeps=20-30	3.71	30
maxeps=40-50	4.9	50

- CPU: core i7 2640M (2.8GHz)
- OS: ubuntu 10.04 LTS (64bit)
- software: GNU C++, boost.interval, boost.ublas

Conclusion

We propose an algorithm to reduce the number of epsilons in affine arithmetic.

- The algorithm is simple and easy to use.
- The algorithm speeds up affine arithmetic.
- The algorithm do not lose the tight inclusion property of affine arithmetic.

Future Work

- Numerical experiments for higher dimensional case.
- Apply to verified IVP solver, especially to very long time integration.

Thank you for your attention!

Conclusion

We propose an algorithm to reduce the number of epsilons in affine arithmetic.

- The algorithm is simple and easy to use.
- The algorithm speeds up affine arithmetic.
- The algorithm do not lose the tight inclusion property of affine arithmetic.

Future Work

- Numerical experiments for higher dimensional case.
- Apply to verified IVP solver, especially to very long time integration.

Thank you for your attention!

Conclusion

We propose an algorithm to reduce the number of epsilons in affine arithmetic.

- The algorithm is simple and easy to use.
- The algorithm speeds up affine arithmetic.
- The algorithm do not lose the tight inclusion property of affine arithmetic.

Future Work

- Numerical experiments for higher dimensional case.
- Apply to verified IVP solver, especially to very long time integration.

Thank you for your attention!

Conclusion

We propose an algorithm to reduce the number of epsilons in affine arithmetic.

- The algorithm is simple and easy to use.
- The algorithm speeds up affine arithmetic.
- The algorithm do not lose the tight inclusion property of affine arithmetic.

Future Work

- Numerical experiments for higher dimensional case.
- Apply to verified IVP solver, especially to very long time integration.

Thank you for your attention!

Conclusion

We propose an algorithm to reduce the number of epsilons in affine arithmetic.

- The algorithm is simple and easy to use.
- The algorithm speeds up affine arithmetic.
- The algorithm do not lose the tight inclusion property of affine arithmetic.

Future Work

- Numerical experiments for higher dimensional case.
- Apply to verified IVP solver, especially to very long time integration.

Thank you for your attention!

Conclusion

We propose an algorithm to reduce the number of epsilons in affine arithmetic.

- The algorithm is simple and easy to use.
- The algorithm speeds up affine arithmetic.
- The algorithm do not lose the tight inclusion property of affine arithmetic.

Future Work

- Numerical experiments for higher dimensional case.
- Apply to verified IVP solver, especially to very long time integration.

Thank you for your attention!

Appendix - In the case of Euclidean norm

For vector $v = (a_1, \cdots, a_p)^T$ we can give penalty function P for Euclidean norm as follows:
Let $S = \{1, 2, \ldots, p\}$ be an index set, 2^S be a set of all subsets of S, then penalty function is give by

$$P(v) = \sqrt{\frac{\max\limits_{S' \in 2^S} \left(\sum\limits_{i \in S'} a_i^2\right)\left(\sum\limits_{i \in S - S'} a_i^2\right)}{\sum\limits_{i \in S} a_i^2}}.$$

To maximize this, we should separate $a_1^2, a_2^2, \ldots, a_p^2$ into two groups which maximize $\left(\sum\limits_{i \in S'} a_i^2\right)\left(\sum\limits_{i \in S - S'} a_i^2\right)$. Namely we should separate into two groups such that the difference of sum of each group becomes as equal as possible. This problem is known as the Number Partitioning Problem, which is NP-complete.
So, we consider the maximum norm version of penalty function is suitable for our algorithm.