Searching for sinks of Hénon map using a multiple-precision GPU arithmetic library

Mioara Joldeș, Valentina Popescu, Warwick Tucker

RAIM 2013

Hénon attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

Hénon attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

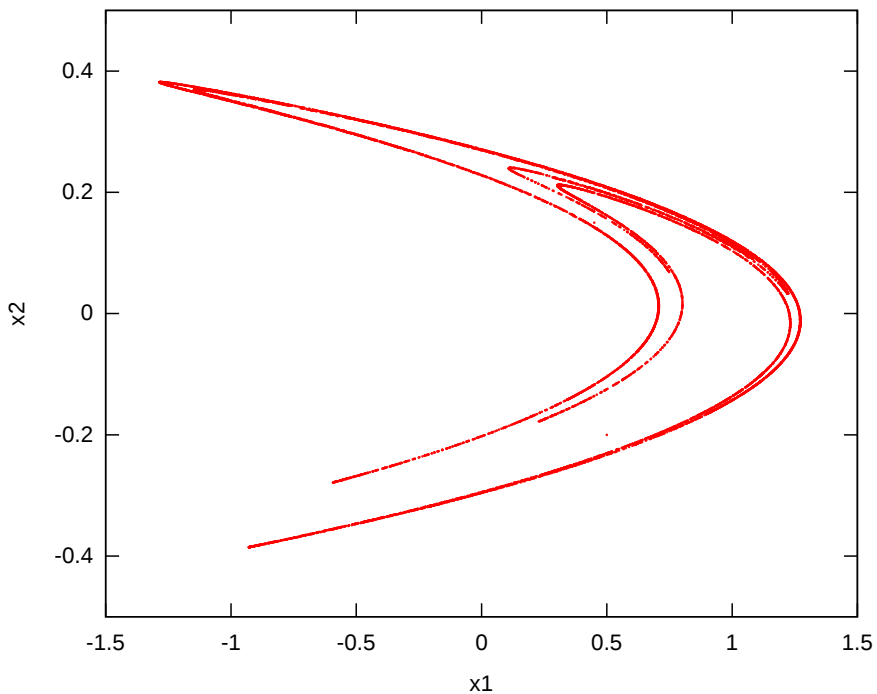
- Map iterations: $h_{a,b}^{i+1} := h_{a,b} \circ h_{a,b}^{i}, i \in \mathbb{N}^*$.

Hénon attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

- Map iterations: $h_{a,b}^{i+1} := h_{a,b} \circ h_{a,b}^{i}, i \in \mathbb{N}^*$.
- For classical parameter values $a = 1.4$, $b = 0.3$ one observes the so-called Hénon attractor by iterating $h_{a,b}^{n}, n \to \infty$:

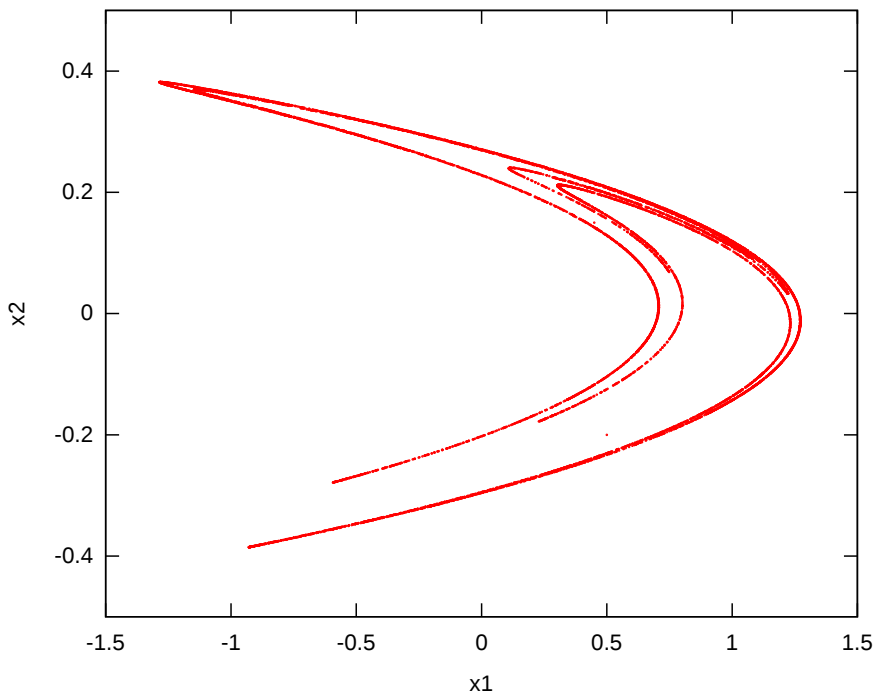


Hénon attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

- Map iterations: $h_{a,b}^{i+1} := h_{a,b} \circ h_{a,b}^{i}, i \in \mathbb{N}^*$.
- For classical parameter values $a = 1.4$, $b = 0.3$ one observes the so-called Hénon attractor by iterating $h_{a,b}^{n}, n \to \infty$:



Open question: Is this a Strange Attractor ?

Some basic terminology

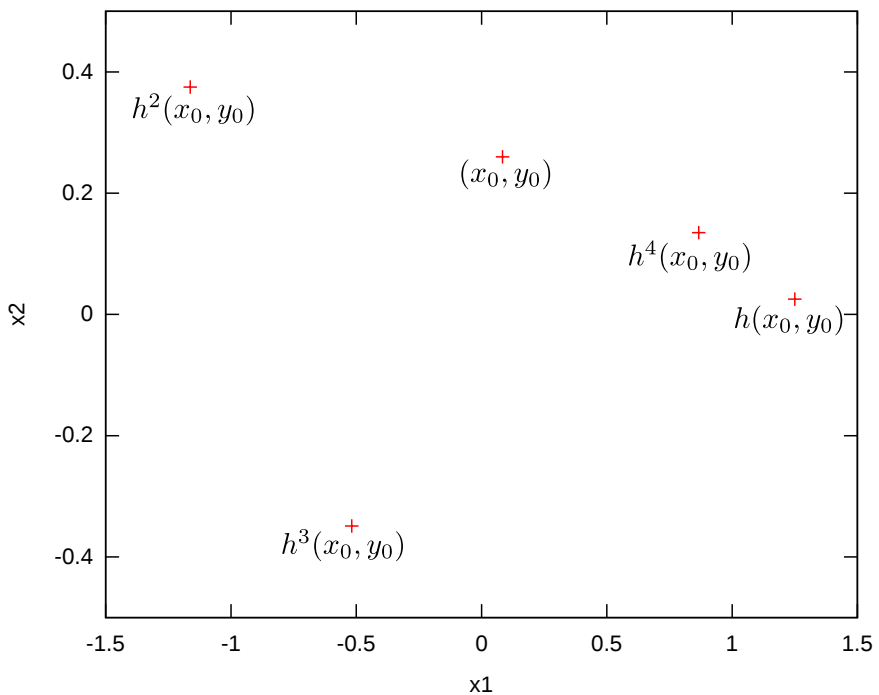
Orbit of x: $\Gamma(x) = \Gamma^+(x) \cup \Gamma^-(x)$

- Forward orbit:

$$\Gamma^+(x) := \{h^n_{a,b}(x), n \in \mathbb{N}\}$$

- Backward orbit:

$$\Gamma^-(x) := \{y : \exists n \in \mathbb{N} : h^n_{a,b}(y) = x\}$$



Some basic terminology

Orbit of x: $\Gamma(x) = \Gamma^+(x) \cup \Gamma^-(x)$

- Forward orbit:
$$\Gamma^+(x) := \{h_{a,b}^n(x), n \in \mathbb{N}\}$$
- Backward orbit:
$$\Gamma^-(x) := \{y : \exists n \in \mathbb{N} : h_{a,b}^n(y) = x\}$$

Periodic orbit $\Gamma_n(x)$ of period n if $\exists n \in \mathbb{N}^*$ s.t. $h_{a,b}^n(x) = x$.

Some basic terminology

Orbit of x: $\Gamma(x) = \Gamma^+(x) \cup \Gamma^-(x)$

- Forward orbit:
$$\Gamma^+(x) := \{h_{a,b}^n(x), n \in \mathbb{N}\}$$
- Backward orbit:
$$\Gamma^-(x) := \{y : \exists n \in \mathbb{N} : h_{a,b}^n(y) = x\}$$

Periodic orbit $\Gamma_n(x)$ of period n if $\exists n \in \mathbb{N}^*$ s.t. $h_{a,b}^n(x) = x$.

Stable orbit $\Gamma(x)$

Let $d(x, y) = ||x - y||$ and $d(y, \Gamma(x)) = \inf_{z \in \Gamma(x)} d(y, z)$.

$\Gamma(x)$ is stable if given $\varepsilon > 0, \exists \delta > 0$ s.t. $d(h_{a,b}^n(y), \Gamma(x)) < \varepsilon$, $\forall n \in \mathbb{N}^*$ and $\forall y$ s.t. $d(y, \Gamma(x)) < \delta$.

Some basic terminology

Orbit of x: $\Gamma(x) = \Gamma^+(x) \cup \Gamma^-(x)$

- Forward orbit:
$$\Gamma^+(x) := \{h^n_{a,b}(x), n \in \mathbb{N}\}$$
- Backward orbit:
$$\Gamma^-(x) := \{y : \exists n \in \mathbb{N} : h^n_{a,b}(y) = x\}$$

Periodic orbit $\Gamma_n(x)$ of period n if $\exists n \in \mathbb{N}^*$ s.t. $h^n_{a,b}(x) = x$.

Stable orbit $\Gamma(x)$

Let $d(x, y) = ||x - y||$ and $d(y, \Gamma(x)) = \inf_{z \in \Gamma(x)} d(y, z)$.

$\Gamma(x)$ is stable if given $\varepsilon > 0, \exists \delta > 0$ s.t. $d(h^n_{a,b}(y), \Gamma(x)) < \varepsilon$, $\forall n \in \mathbb{N}^*$ and $\forall y$ s.t. $d(y, \Gamma(x)) < \delta$.

Asymptotically Stable orbit $\Gamma(x)$ (sink)

$\Gamma(x)$ is asymptotically stable if it is stable and (by choosing δ smaller if necessary) $d(h^n_{a,b}(y), \Gamma(x)) \to 0$ as $n \to \infty$.

Some basic terminology II

Attractor

- describes asymptotic behaviour of typical orbits.

Some basic terminology II

Attractor

- describes asymptotic behaviour of typical orbits.
- Trapped Attracting Sets

  Let T be a compact set such that $h(T) \subset \mathrm{int}(T)$. Then $A = \bigcap_{i=0}^{\infty} h^i(T)$ is a trapped attracting set.

Some basic terminology II

Attractor

- describes asymptotic behaviour of typical orbits.
- Trapped Attracting Sets

  Let T be a compact set such that $h(T) \subset \mathrm{int}(T)$. Then $A = \bigcap_{i=0}^{\infty} h^i(T)$ is a trapped attracting set.
- Attractor: An attracting set which contains a dense orbit.

Some basic terminology II

Attractor

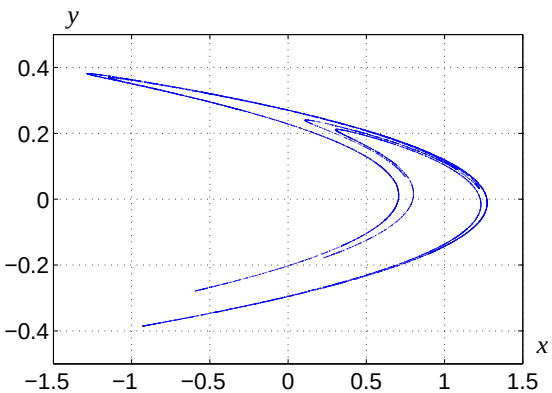
- describes asymptotic behaviour of typical orbits.
- Trapped Attracting Sets

  Let T be a compact set such that $h(T) \subset \mathrm{int}(T)$. Then $A = \bigcap_{i=0}^{\infty} h^i(T)$ is a trapped attracting set.
- Attractor: An attracting set which contains a dense orbit.

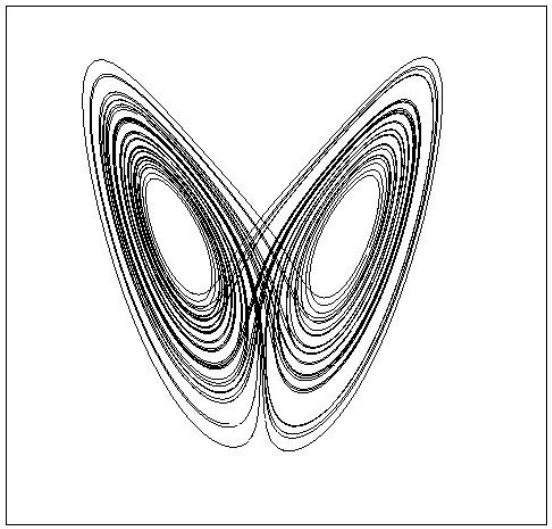
Basin of attraction

Union of orbits which converge towards the attractor.

Some basic terminology III - Strange Attractor



(a) Hénon Attractor

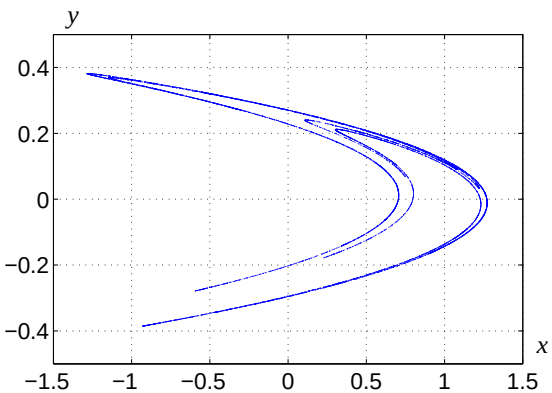


(b) Lorenz Attractor

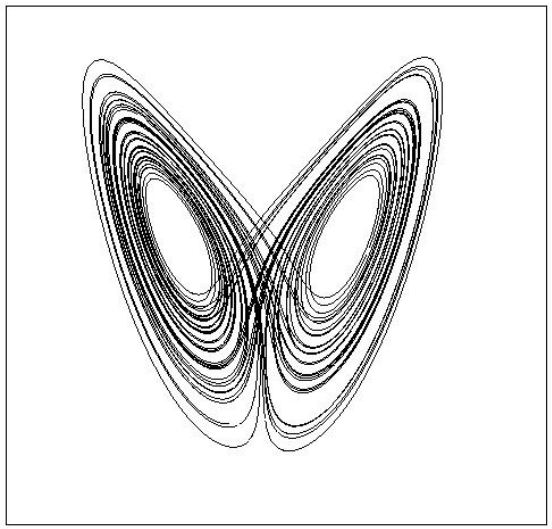
(c) Ueda Attractor

- name coined by Takens and Ruelle $\simeq 1971$

Some basic terminology III - Strange Attractor



(a) Hénon Attractor

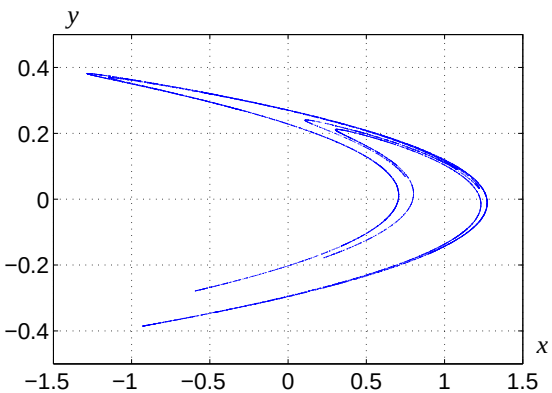


(b) Lorenz Attractor

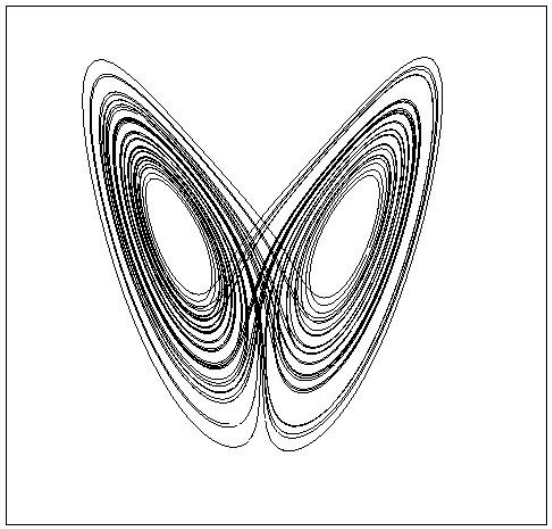
(c) Ueda Attractor

- name coined by Takens and Ruelle $\simeq 1971$

Some basic terminology III - Strange Attractor



(a) Hénon Attractor



(b) Lorenz Attractor

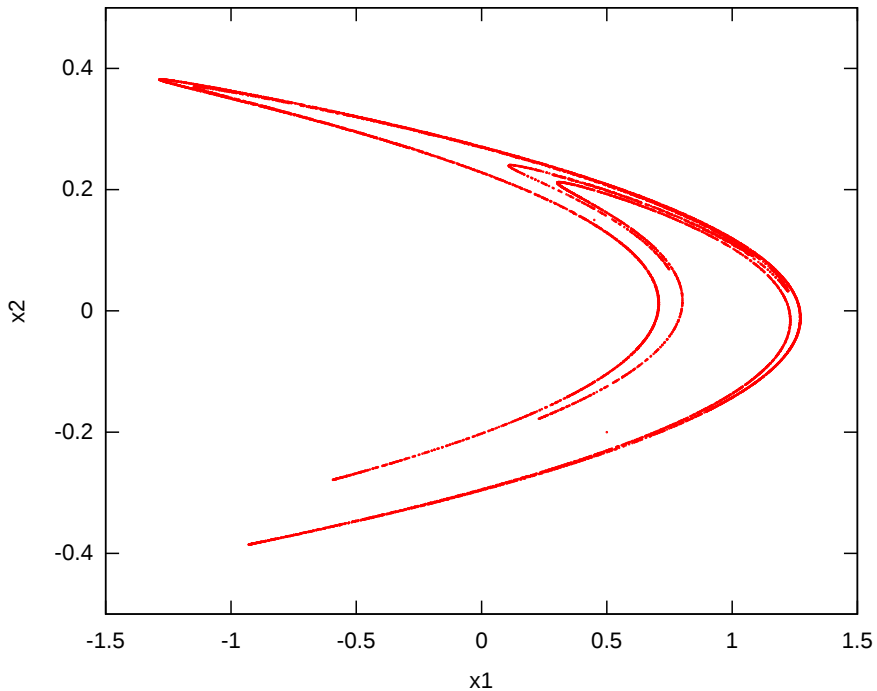
(c) Ueda Attractor

- name coined by Takens and Ruelle $\simeq 1971$
- Properties:[Ruelle 2006]
  - chaotic dynamics e.g.
  - fractal dimension fig.
  - invariant in the sense that the system comes arbitrarily close to any point on the attractor independently of the initial conditions.

Hénon Attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

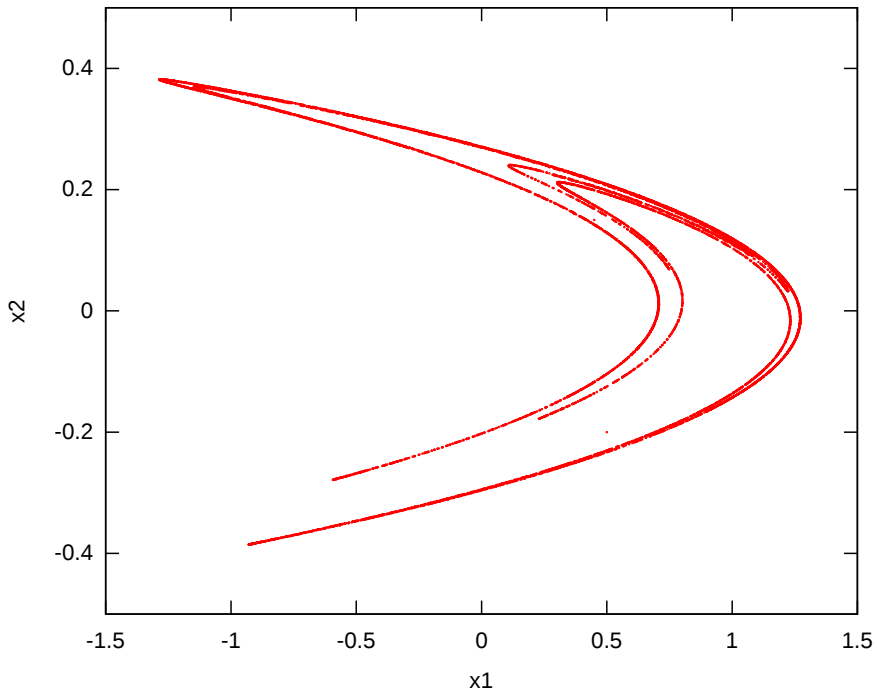


Open question: Is this a Strange Attractor ?

Hénon Attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$



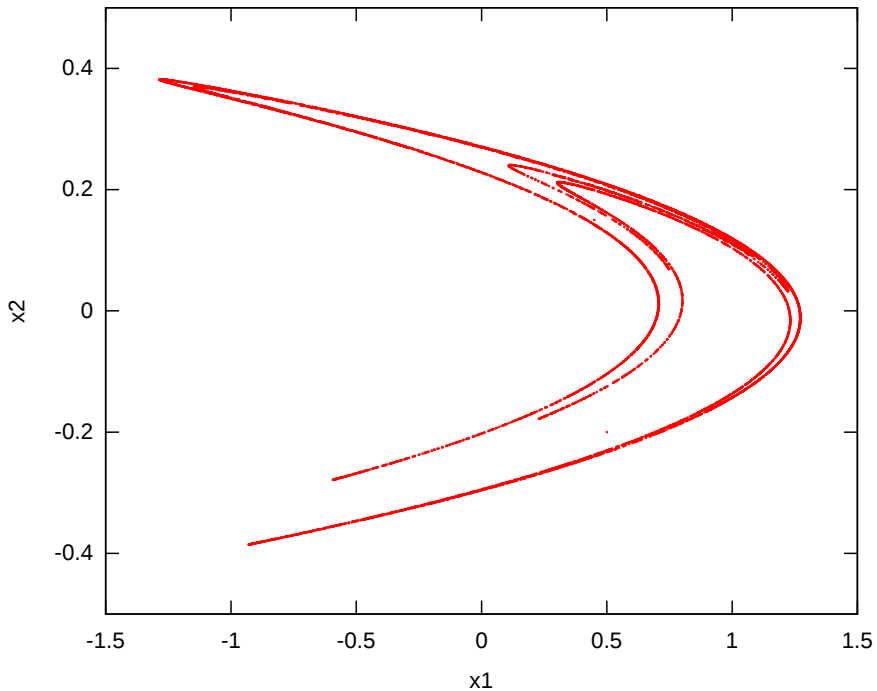
Open question: Is this a Strange Attractor ?

- Chaotic map: aperiodic trajectories (generally believed)

Hénon Attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$



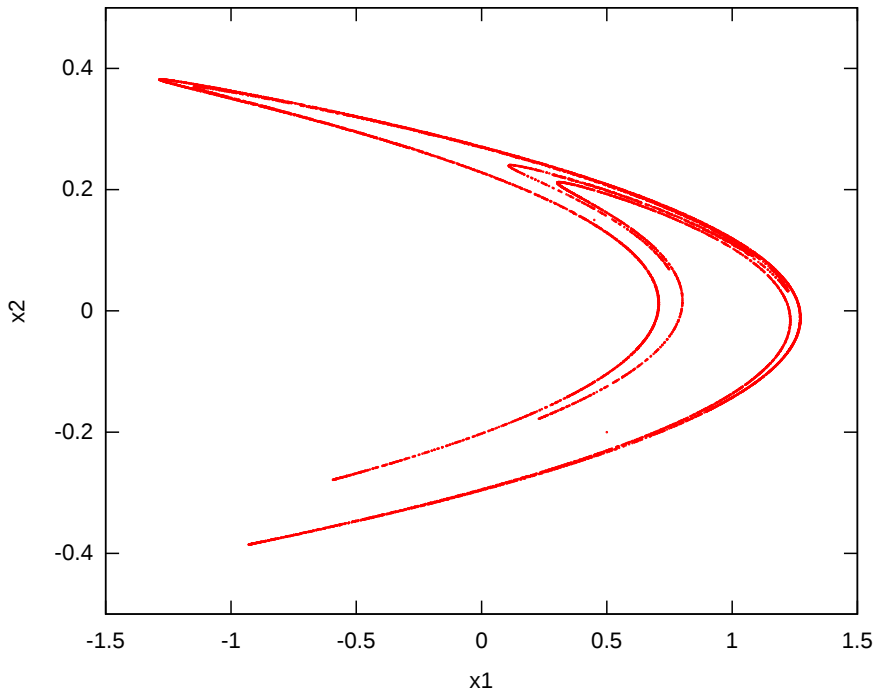
Open question: Is this a Strange Attractor ?

- Chaotic map: aperiodic trajectories (generally believed)
- there is a set of parameters (near $b = 0$) with positive Lebesgue measure for which the Hénon map has a strange attractor. [Benedicks & Carleson,'91].

Hénon Attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$



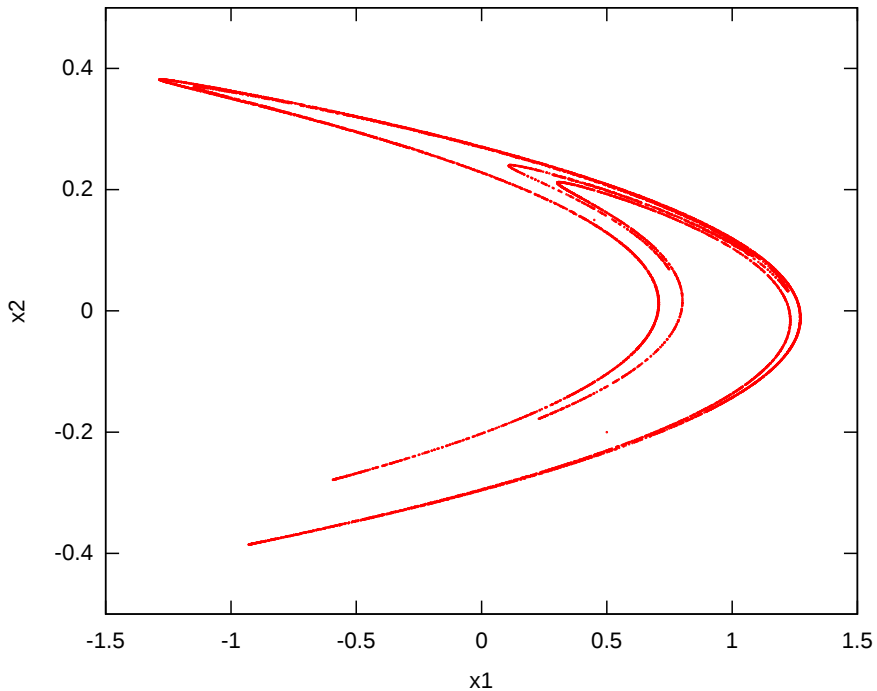
Open question: Is this a Strange Attractor ?

- Chaotic map: aperiodic trajectories (generally believed)
- there is a set of parameters (near $b = 0$) with positive Lebesgue measure for which the Hénon map has a strange attractor. [Benedicks & Carleson,'91].
- this set is believed to be densely filled with regions, where the attractor is periodic.

Hénon Attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$



Open question: Is this a Strange Attractor ?

- Chaotic map: aperiodic trajectories (generally believed)
- there is a set of parameters (near $b = 0$) with positive Lebesgue measure for which the Hénon map has a strange attractor. [Benedicks & Carleson,'91].
- this set is believed to be densely filled with regions, where the attractor is periodic.

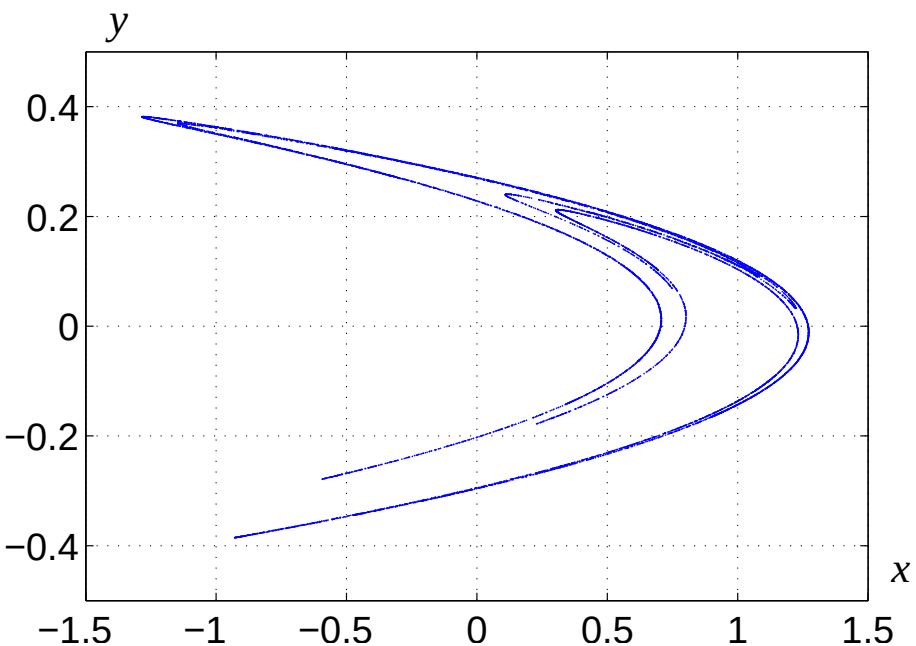
Goal: Given (a, b), prove existence of sinks (stable periodic orbits).

Example of sink in Hénon attractor [Galias & Tucker 2013]

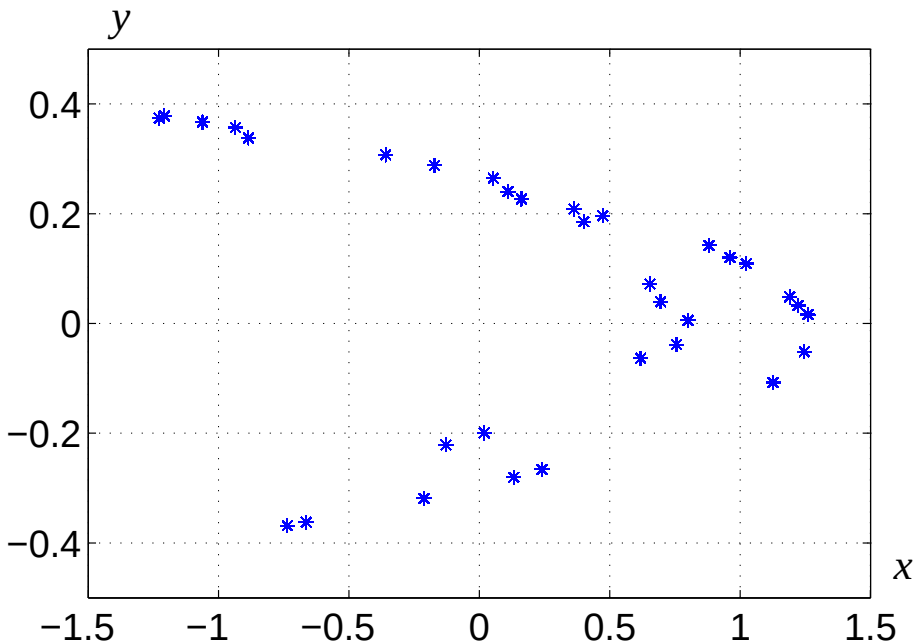
Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

Let $a = 1.399999486944$, $b = 0.3$. Trajectory composed of 10000 points:



(a) $5 \cdot 10^9$ iterations skipped



(b) $6 \cdot 10^9$ iterations skipped.

Hénon attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

Goal: Given (a, b), prove existence of sinks (stable periodic orbits).

Method:

Hénon attractor

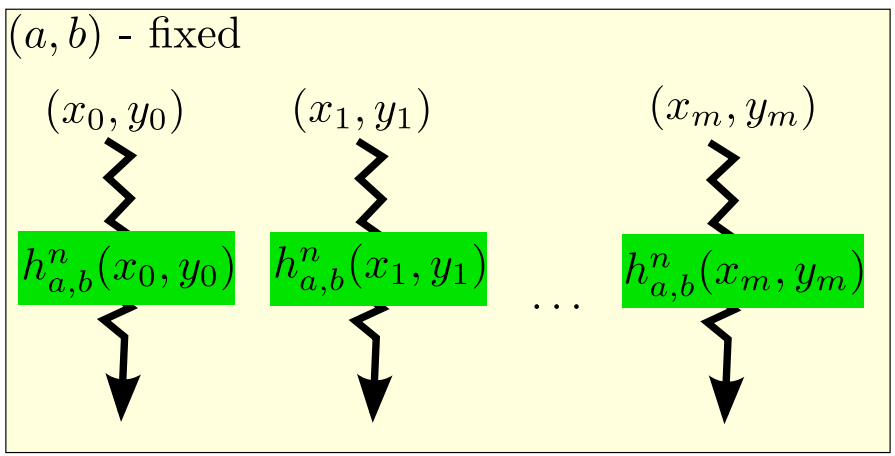
Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

Goal: Given (a, b), prove existence of sinks (stable periodic orbits).

Method:

1. Find numerical approximation of sinks
   - many initial conditions($m \sim 2^9 \dots 2^{20}$); many iterations($n \sim 10^6 \dots 10^9$)/parameter



Hénon attractor

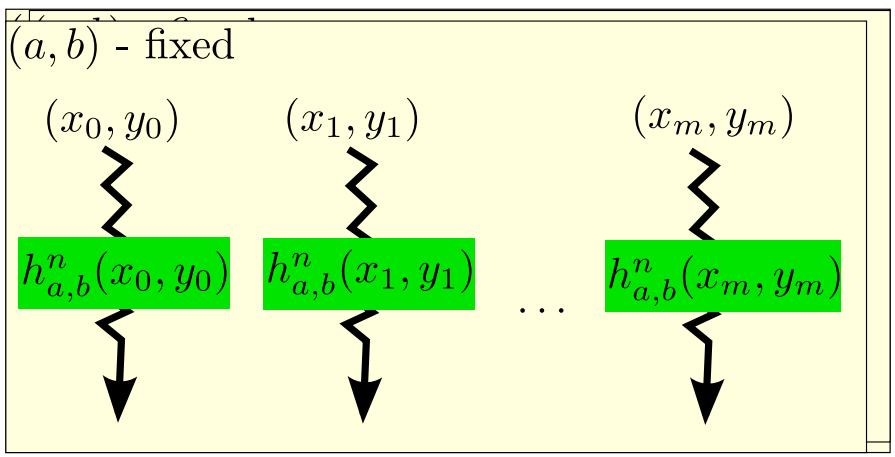
Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

Goal: Given (a, b), prove existence of sinks (stable periodic orbits).

Method:

1. Find numerical approximation of sinks
   - many initial conditions($m \sim 2^9 \dots 2^{20}$); many iterations($n \sim 10^6 \dots 10^9$)/parameter



Hénon attractor

Hénon Map

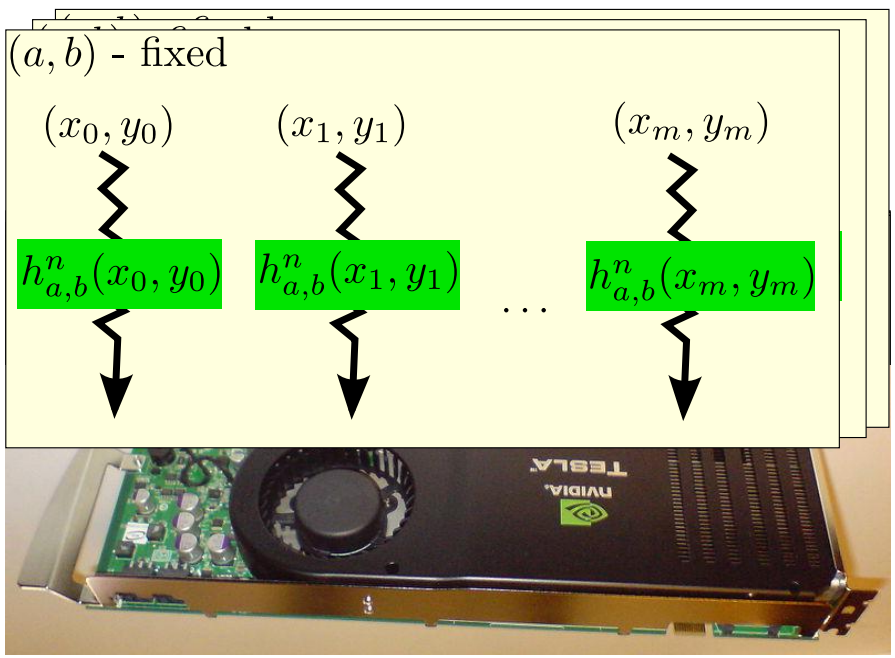
$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

Goal: Given (a, b), prove existence of sinks (stable periodic orbits).

Method:

1. Find numerical approximation of sinks
   - many initial conditions($m \sim 2^9 \dots 2^{20}$); many iterations($n \sim 10^6 \dots 10^9$)/parameter

High parallelism $\rightarrow$
Graphics Processing Units
(GPUs)



Hénon attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

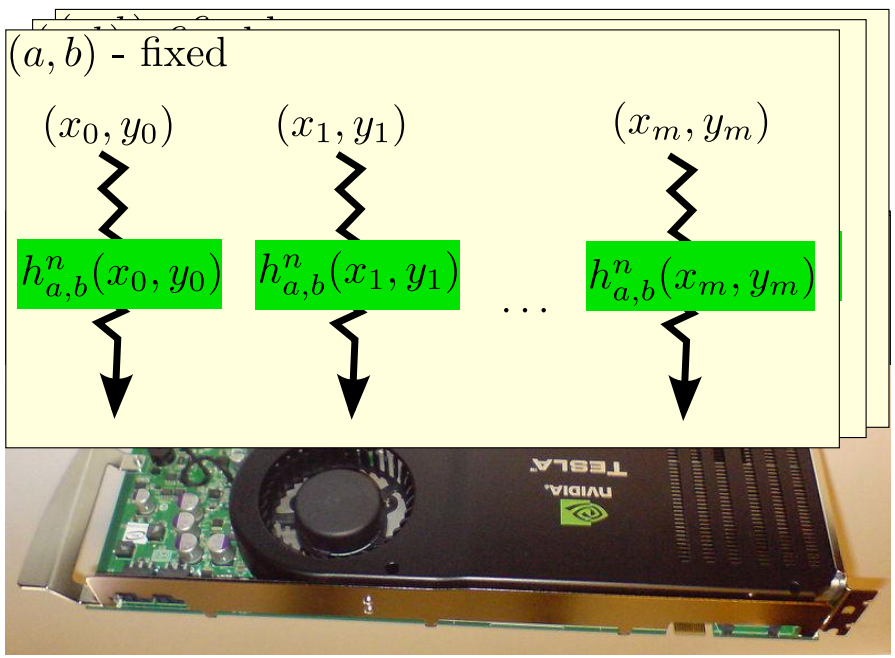
Goal: Given (a, b), prove existence of sinks (stable periodic orbits).

Method:

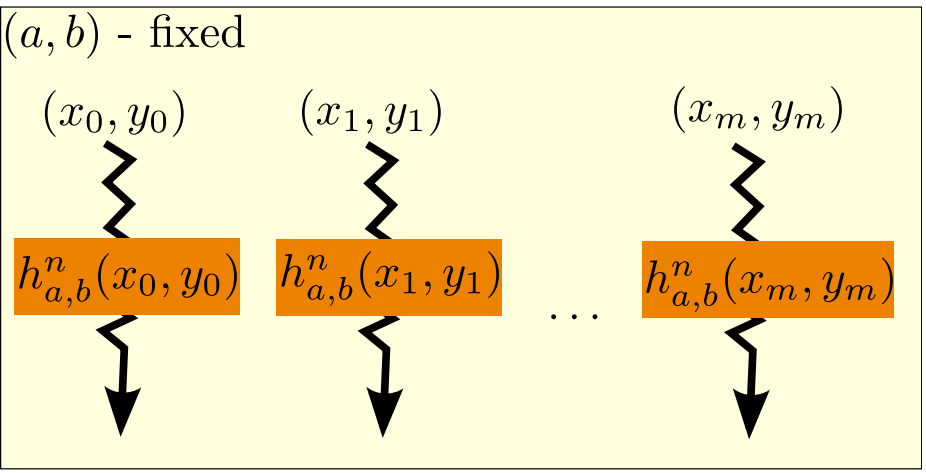
1. Find numerical approximation of sinks
   - many initial conditions($m \sim 2^9 \dots 2^{20}$); many iterations($n \sim 10^6 \dots 10^9$)/parameter
   - chaotic $\rightarrow$ multiple precision

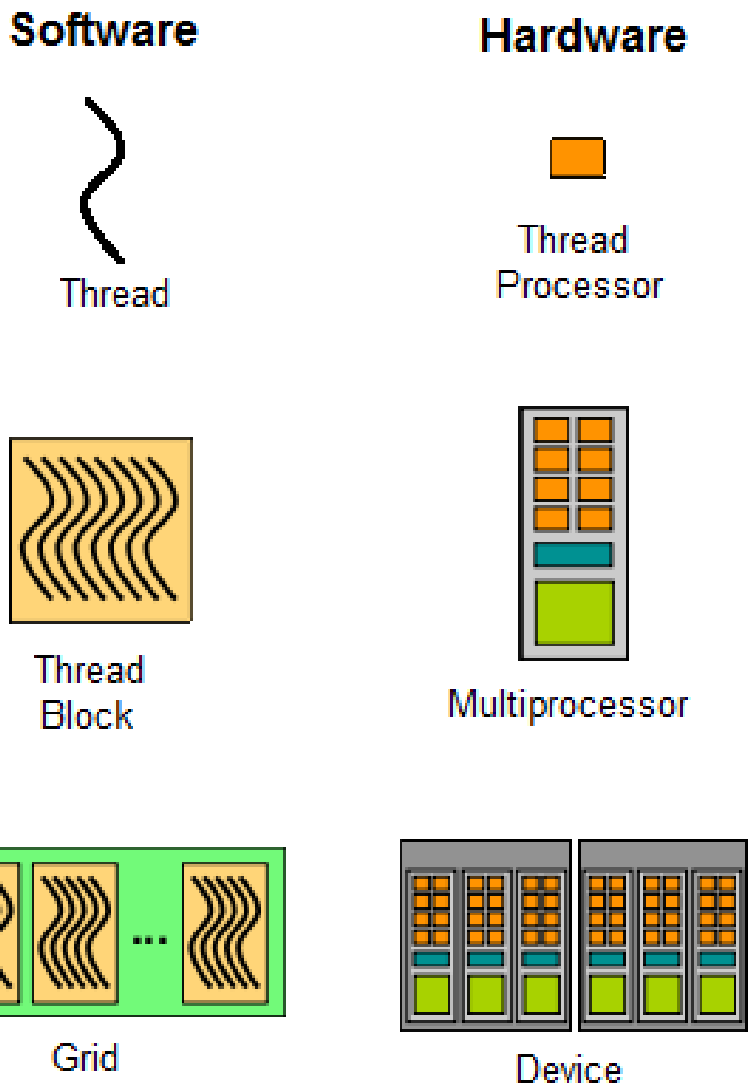
High parallelism $\rightarrow$
Graphics Processing Units
(GPUs)

Need for Multiple
Precison on GPUs



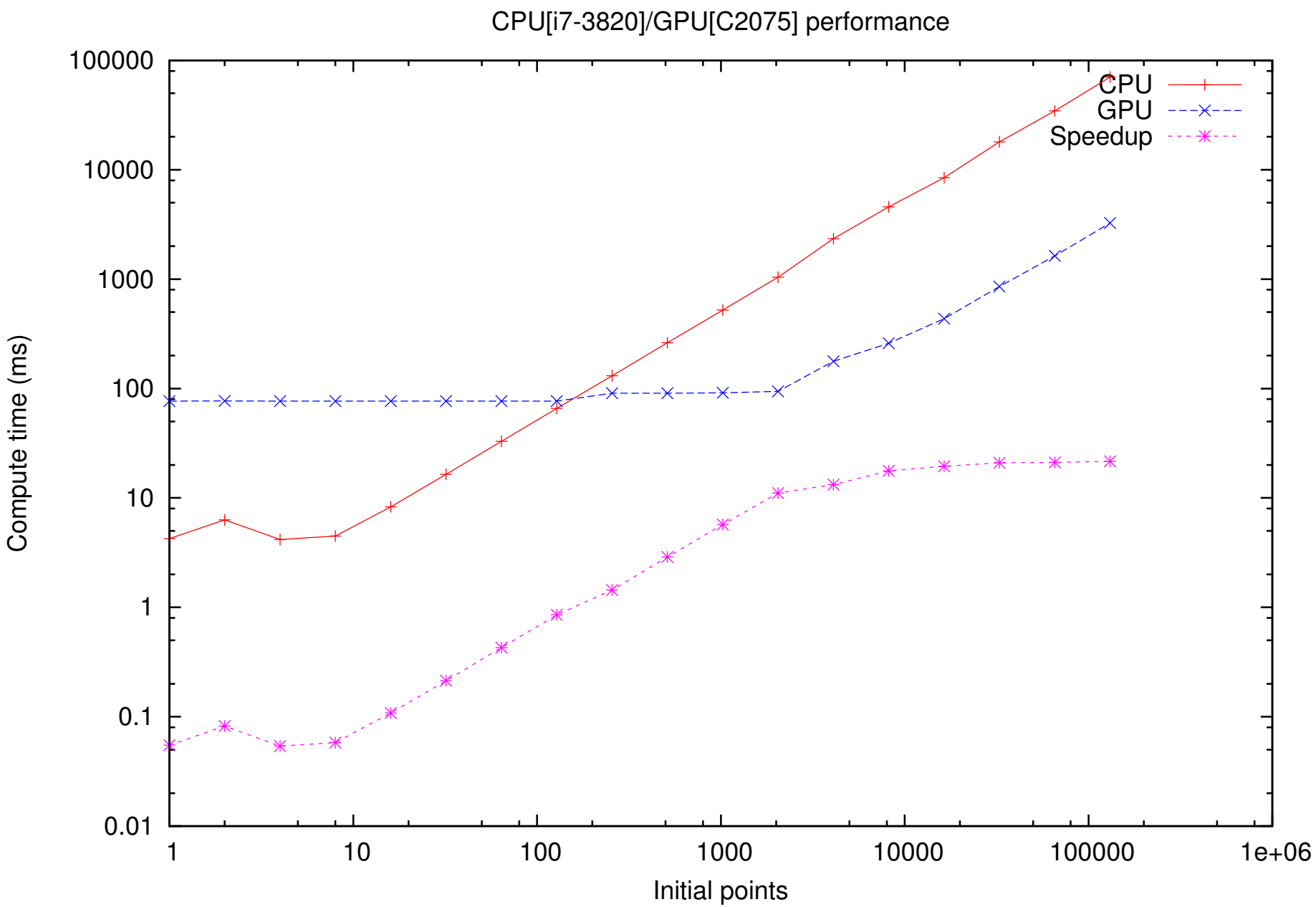
Hénon on GPU - Execution model in a nutshell





Example: Fermi Arch – up to 512 thread processors in 16 multiprocessors

Hénon on GPU - Speed-up



Hénon attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

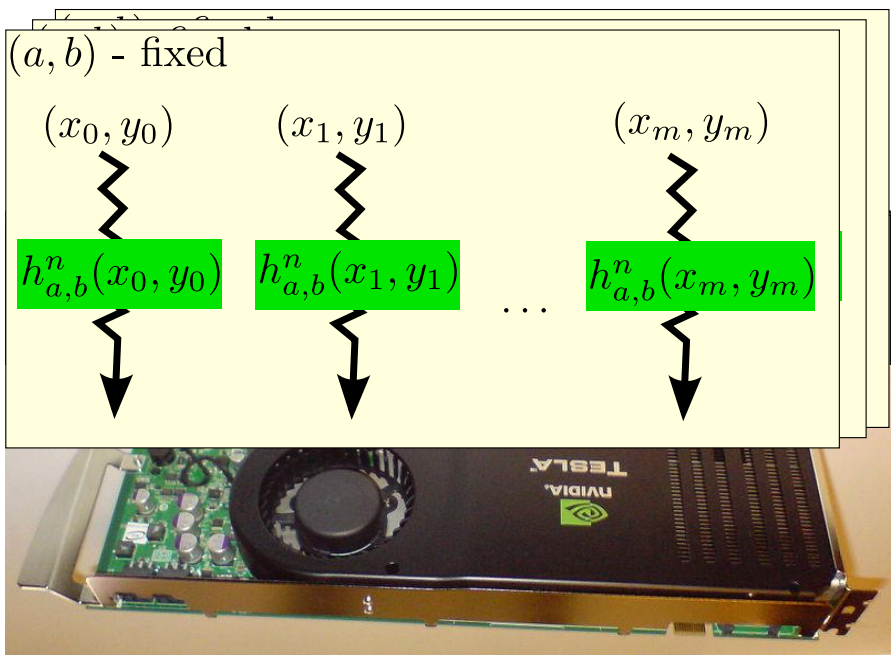
Goal: Given (a, b), prove existence of sinks (stable periodic orbits).

Method:

1. Find numerical approximation of sinks
   - many initial conditions($m \sim 2^9 \dots 2^{20}$); many iterations($n \sim 10^6 \dots 10^9$)/parameter
   - chaotic $\rightarrow$ multiple precision

High parallelism $\rightarrow$
Graphics Processing Units
(GPUs)

Need for Multiple
Precison on GPUs



Hénon attractor

Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$

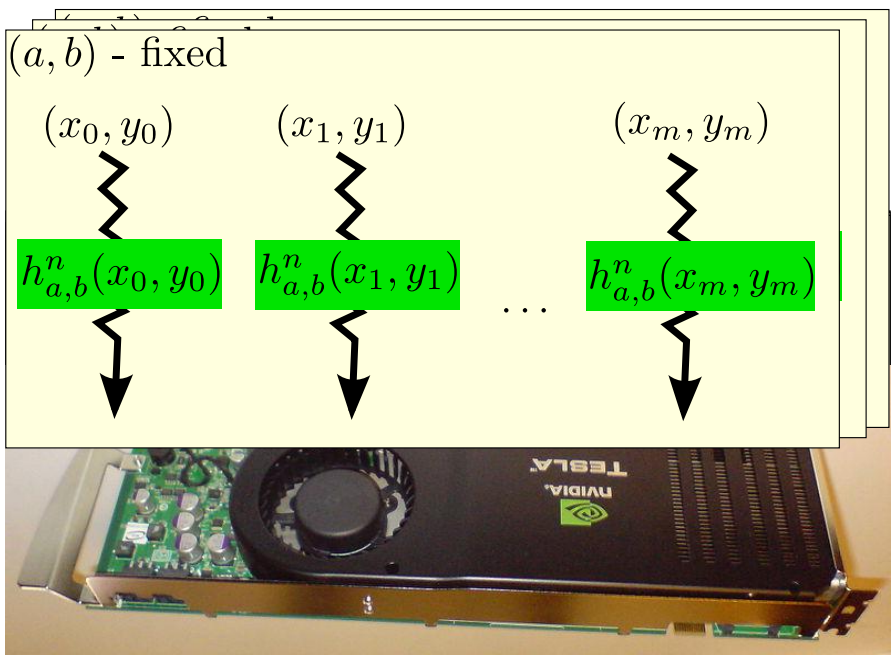
Goal: Given (a, b), prove existence of sinks (stable periodic orbits).

Method:

1. Find numerical approximation of sinks
   - many initial conditions($m \sim 2^9 \dots 2^{20}$); many iterations($n \sim 10^6 \dots 10^9$)/parameter
   - chaotic $\rightarrow$ multiple precision
2. A posteriori validation of existence and stability with interval arithmetic

High parallelism $\rightarrow$
Graphics Processing Units
(GPUs)

Need for Multiple
Precison on GPUs



Rigorous a posteriori proof of existence: interval Newton operator [Neumaier 1990]

Theorem

Let $F : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$, $F \in \mathcal{C}^1(D)$, $\boldsymbol{v} \in \mathbb{ID}$, $\hat{v} \in \boldsymbol{v}$.

$$N(\boldsymbol{v}) := \hat{v} - F'(\boldsymbol{v})^{-1} F(\hat{v})$$

If $N(\boldsymbol{v})$ is well defined, then the following statements hold:

(1) if $\boldsymbol{v}$ contains a zero x^* of F, then so does $N(\boldsymbol{v}) \cap \boldsymbol{v}$;

(2) if $N(\boldsymbol{v})) \cap \boldsymbol{v} = \emptyset$, then $\boldsymbol{v}$ has no zeros of F;

(3) if $N(\boldsymbol{v})) \subseteq \boldsymbol{v}$, then $\boldsymbol{v}$ contains a unique zero of F;

Proof.

(1) Follows from Mean Value Theorem;

(2) Contra-positive of (1);

(3) Existence from Brouwer's fixed point theorem; uniqueness from non-vanishing F'.

□

$\mathbb{ID}$ is the set of all real intervals included in D.

Validation of periodic orbits using Newton operator [Galias 2001], [Galias & Tucker, 2011, 2013]

- Let $v = (z_0, z_1, \ldots, z_{p-1})$ be the computer generated period p orbit.

Validation of periodic orbits using Newton operator [Galias 2001], [Galias & Tucker, 2011, 2013]

- Let $v = (z_0, z_1, \ldots, z_{p-1})$ be the computer generated period p orbit.

Remark

Let $v = (z_0, z_1, \ldots, z_{p-1})$ and

$$[F(v)]_k = z_{(k+1) \bmod p} - h(z_k), k = 0, \ldots, p-1.$$

Then $F(v) = 0$ iif $h^p(z_0) = z_0$.

$\longrightarrow$ existence of zeros of $\mathrm{Id} - h^p \iff$ existence of zeros of F;
$\longrightarrow$ reduced overestimation using F [Galias 2001].

Validation of periodic orbits using Newton operator [Galias 2001], [Galias & Tucker, 2011, 2013]

- Let $v = (z_0, z_1, \ldots, z_{p-1})$ be the computer generated period p orbit.

Remark

Let $v = (z_0, z_1, \ldots, z_{p-1})$ and

$$[F(v)]_k = z_{(k+1) \bmod p} - h(z_k), k = 0, \ldots, p-1.$$

Then $F(v) = 0$ iif $h^p(z_0) = z_0$.

$\longrightarrow$ existence of zeros of $\mathrm{Id} - h^p \iff$ existence of zeros of F;
$\longrightarrow$ reduced overestimation using F [Galias 2001].

- Choose $r > 0$ s.t. $\boldsymbol{v} = ([z_0 - r, z_0 + r], \ldots, [z_{p-1} - r, z_{p-1} + r])$.

Validation of periodic orbits using Newton operator [Galias 2001], [Galias & Tucker, 2011, 2013]

- Let $v = (z_0, z_1, \ldots, z_{p-1})$ be the computer generated period p orbit.

Remark

Let $v = (z_0, z_1, \ldots, z_{p-1})$ and

$$[F(v)]_k = z_{(k+1) \bmod p} - h(z_k), k = 0, \ldots, p-1.$$

Then $F(v) = 0$ iif $h^p(z_0) = z_0$.

$\longrightarrow$ existence of zeros of $\mathrm{Id} - h^p$ $\iff$ existence of zeros of F;
$\longrightarrow$ reduced overestimation using F [Galias 2001].

- Choose $r > 0$ s.t. $\boldsymbol{v} = ([z_0 - r, z_0 + r], \ldots, [z_{p-1} - r, z_{p-1} + r])$.
- Verify that $N(\boldsymbol{v}) \subseteq \boldsymbol{v}$.

Floating-Point Multiple Precision on Graphics Processing Units (GPUs)

High parallelism + Multiple precision $\rightarrow$ Need for Multiple Precison on GPUs

Floating-Point Multiple Precision on Graphics Processing Units (GPUs)

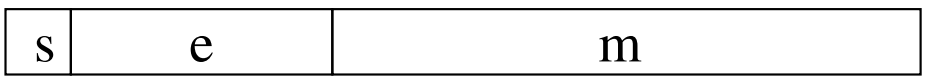
High parallelism + Multiple precision $\rightarrow$ Need for Multiple Precison on GPUs

High parallelism + Multiple precision $\rightarrow$ Need for Multiple Precison on GPUs

1. Floating Point (FP): A real number X is approximated in a machine by a rational $x = (-1)^s \times m \times \beta^e$
   - β is the radix (usually 2)
   - s is the sign bit
   - m is the mantissa, a rational number of p digits in radix β

     $$m = d_0.d_1 d_2 \dots d_{p-1}$$

   - e is the exponent, a signed integer on n_e bits

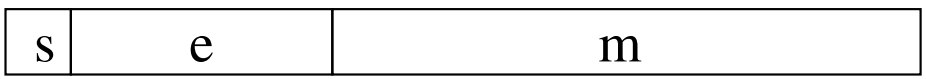


High parallelism + Multiple precision → Need for Multiple Precison on GPUs

1. Floating Point (FP): A real number X is approximated in a machine by a rational $x = (-1)^s \times m \times \beta^e$
   - β is the radix (usually 2)
   - s is the sign bit
   - m is the mantissa, a rational number of p digits in radix β

     $$m = d_0.d_1 d_2 \dots d_{p-1}$$

   - e is the exponent, a signed integer on n_e bits



2. GPUs:
   - used for scientific computing: GP-GPU programs
   - Languages: OpenCL, CUDA (NVIDIA)
   - conformant with IEEE-754 standard
   - Support for all rounding modes for basic operations $(+, -, *, /, \sqrt{})$
   - Dynamic rounding mode without any penalties:

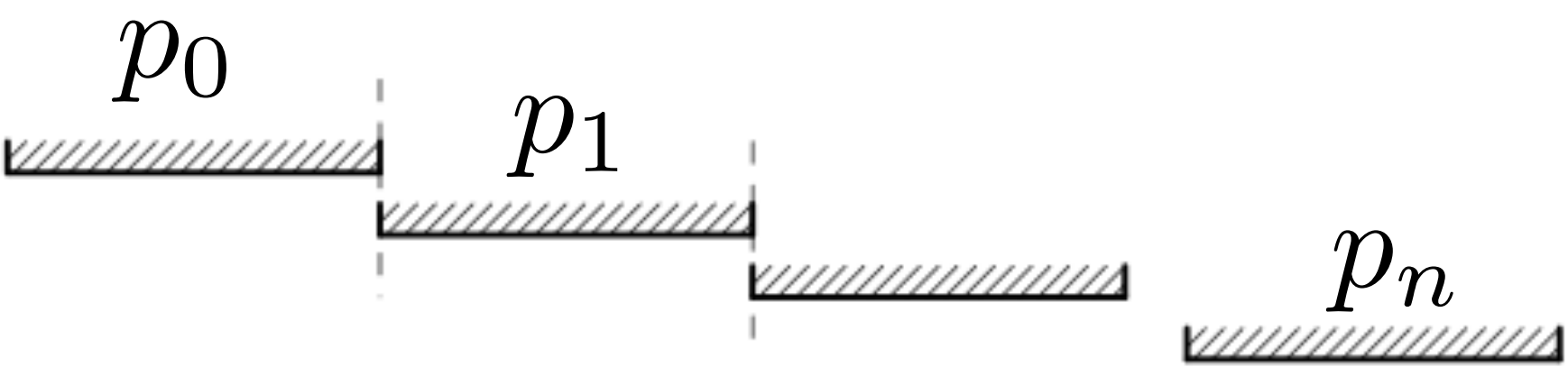
     `__dadd_rn(a,b)=RN(a+b)`
   - The FMA (Fused Multiply-Add):

     `__dfma_rn (a,b,c)= RN(A *B + C)`

Our approach: multiple-term expansions, i.e.

a number is expressed as a non-evaluated sum of FP numbers
$N = p_0 + \ldots + p_n$.

The expansions are required to be non-overlapping.



Previous works: Zimmer, et al. (C - XSC Library with Kulisch long accumulator), Dekker71, Priest91, Shewchuk96, Bailey01, Nievergelt04, Rump05

Precision-2p vs. double-double

π in double-double

$$p_1 = 11.0010010000111111011010101000100010000101101000110000_2,$$

and

$$p_2 = 1.0001101001100010011000110011000101000101110000000111_2 \times 2^{-53}.$$

$p_1 + p_2 \leftrightarrow 107$ bits FP approx.

Non-overlapping FP Sequences

Precision-2p vs. double-double

π in double-double

$$p_1 = 11.0010010000111111011010101000100010000101101000110000_2,$$

and

$$p_2 = 1.0001101001100010011000110011000101000101110000000111_2 \times 2^{-53}.$$

$p_1 + p_2 \leftrightarrow 107$ bits FP approx.

$\ln(12)$ in double-double

$$\ell_1 = 10.011111000010001011010111100110100111001111001111101_2,$$

and

$$\ell_2 = -1.100110001110010000001111100001011011110101111001011 1_2 \times 2^{-55}.$$

$\ell_1 + \ell_2 \leftrightarrow 109$ bits FP approx.

Error-Free Transformations: 2Sum & 2Prod

Error-Free Transformations: 2Sum & 2Prod

Example 2Sum:

$$
\begin{aligned}
a &= 1.\underbrace{0\ldots0}_{50}11 \times 2^0 \\
b &= 0.\underbrace{0\ldots00}_{51}111 \times 2^0 = 1.11 \times 2^{-52}
\end{aligned}
$$

Error-Free Transformations: 2Sum & 2Prod

Example 2Sum:

$$
\begin{aligned}
a &= 1.\underbrace{0\ldots0}_{50}11 \times 2^0 \\
b &= 0.\underbrace{0\ldots00}_{51}111 \times 2^0 = 1.11 \times 2^{-52} \\
s &= 1.\underbrace{0\ldots0}_{49}101 \times 2^0 \\
t &= -0.\underbrace{0\ldots00}_{53}1 \times 2^0 = -1 \times 2^{-54} \\
a + b = s + t &= 1.\underbrace{0\ldots0}_{49}10011 \times 2^0
\end{aligned}
$$

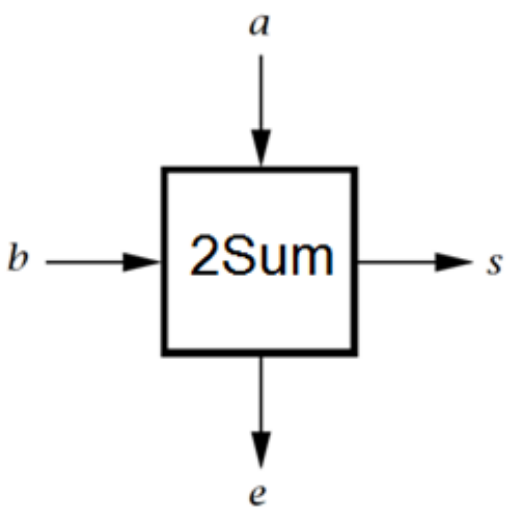
Error-Free Transformations: 2Sum & 2Prod

Assume the floating-point system being used has radix $\beta \leq 3$, subnormal numbers available, and provides correct rounding with rounding to nearest.

Theorem 1 (2Sum algorithm)

Let a and b be floating-point numbers. Algorithm 2Sum computes two floating-point numbers s and t that satisfy the following:

- $s + t = a + b$ *exactly;*
- $s = RN(a + b)$.



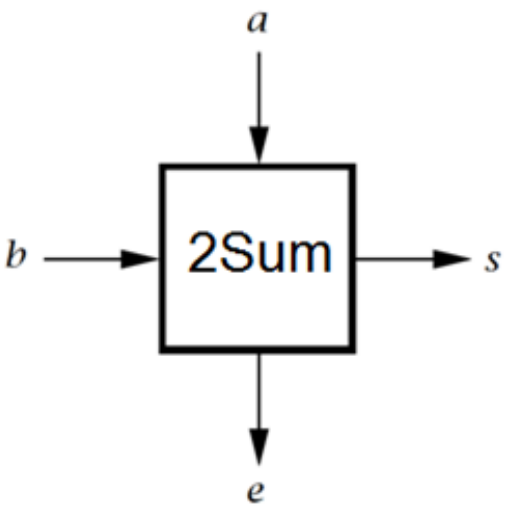
Error-Free Transformations: 2Sum & 2Prod

Assume the floating-point system being used has radix $\beta \leq 3$, subnormal numbers available, and provides correct rounding with rounding to nearest.

Theorem 1 (2Sum algorithm)

Let a and b be floating-point numbers. Algorithm 2Sum computes two floating-point numbers s and t that satisfy the following:

- $s + t = a + b$ *exactly;*
- $s = RN(a + b)$.




```
/* Computes RN(a+b) and t=(a+b)-RN(a+b).  */
__device__ static __forceinline__
  double 2Sum(double a, double b, double &t) {
  double s = __dadd_rn(a,b);
  double aa = __dadd_rn(s,-b);
  double bb = __dadd_rn(s,-aa);
  double da = __dadd_rn(a,-aa);
  double db = __dadd_rn(b,-bb);
  t= __dadd_rn(da,db);
  return s;
}
```

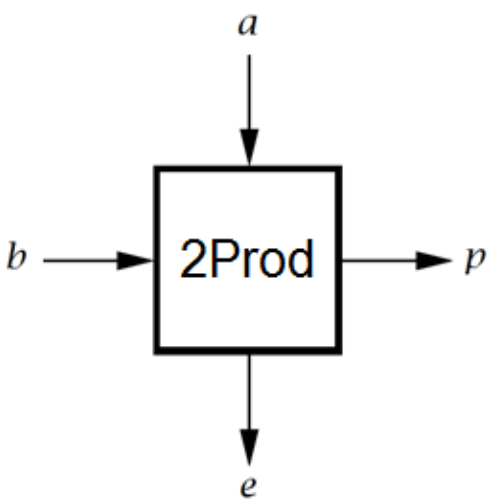
Error-Free Transformations: 2Sum & 2Prod

Assume the floating-point system being used has radix $\beta \leq 3$, subnormal numbers available, and provides correct rounding with rounding to nearest.

Theorem 2 (2ProdFMA algorithm)

Let a and b be floating-point numbers, $e_a + e_b \geq e_{min} + p - 1$. Algorithm 2ProdFMA computes two floating-point numbers p and e that satisfy the following:

- $p + e = a * b$ *exactly;*
- $p =$ *RN*$(a * b)$.



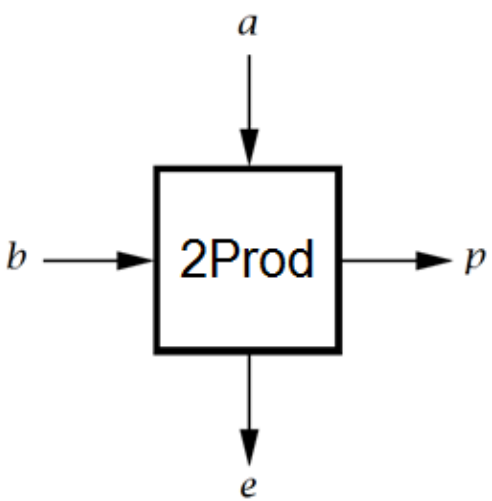
Error-Free Transformations: 2Sum & 2Prod

Assume the floating-point system being used has radix $\beta \leq 3$, subnormal numbers available, and provides correct rounding with rounding to nearest.

Theorem 2 (2ProdFMA algorithm)

Let a and b be floating-point numbers, $e_a + e_b \geq e_{min} + p - 1$. Algorithm 2ProdFMA computes two floating-point numbers p and e that satisfy the following:

- $p + e = a * b$ *exactly;*
- $p = RN(a * b)$.

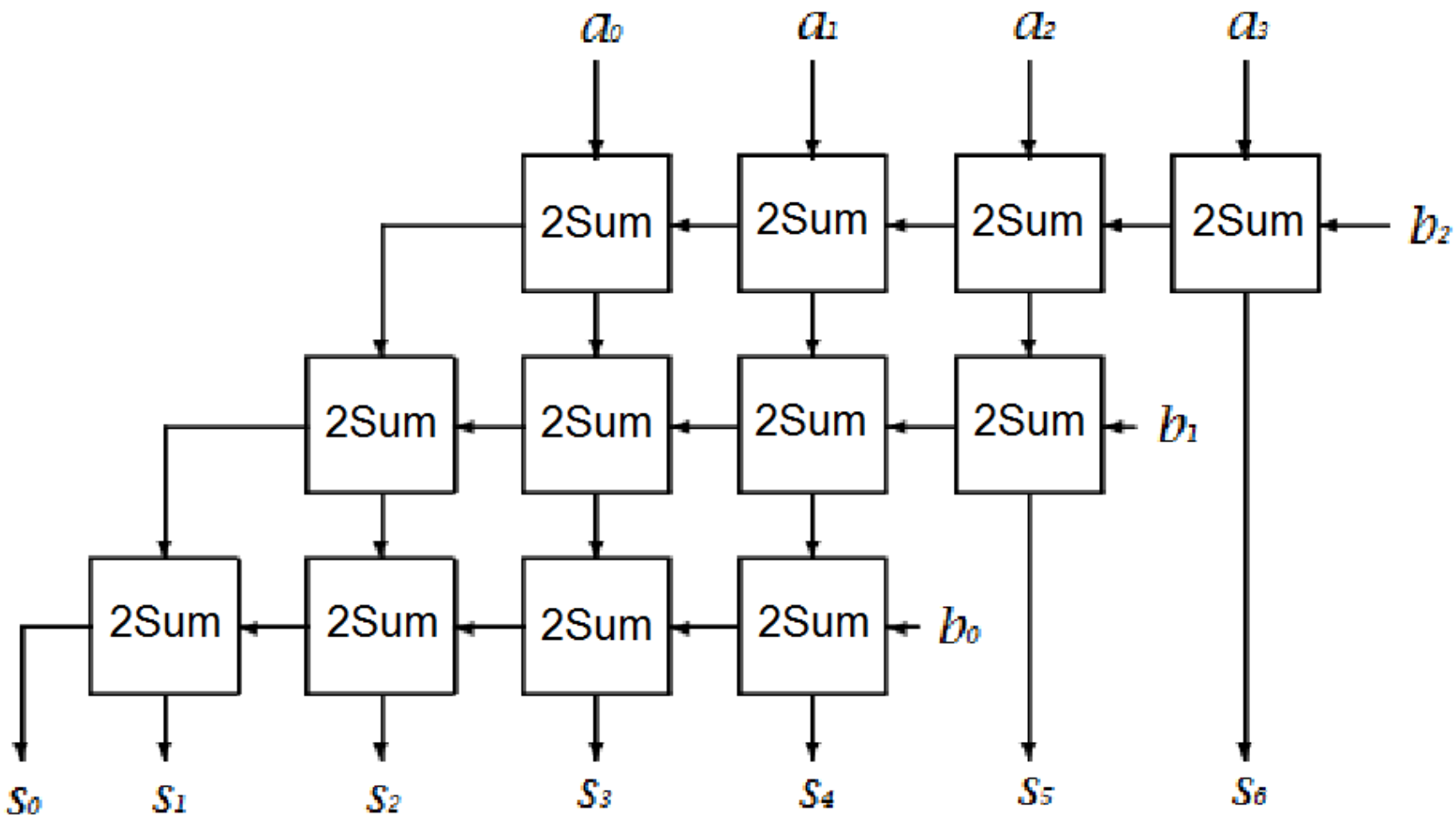



```
/* Computes p=RN(a*b) and e=(a*b)-p, using an FMA.
__device__ static __forceinline__
  double 2ProdFMA(double a, double b, double &e) {
  double p = __dmul_rn(a, b);
  e = __fma_rn(a, b, -p);
  return p;
}
```

Shewchuck's ExpansionSum

Input: (a_i), (b_i) non-overlapping expansions

Output:(s_i) non-overlapping expansion, s.t. $\sum_{i=1}^{n+m} s_i = \sum_{i=1}^{n} a_i + \sum_{i=1}^{m} b_i$



Parallelisation of search for sinks on Haddock cluster

- Haddock
  `http://www.uppmax.uu.se/the-haddock-cluster`
  28 compute nodes with dual Tesla M2075 GPUs.

Parallelisation of search for sinks on Haddock cluster

- Haddock
  `http://www.uppmax.uu.se/the-haddock-cluster`
  28 compute nodes with dual Tesla M2075 GPUs.

  - Each parameter value (a,b) assigned a block.

Parallelisation of search for sinks on Haddock cluster

- Haddock
  `http://www.uppmax.uu.se/the-haddock-cluster`
  28 compute nodes with dual Tesla M2075 GPUs.

- Each parameter value (a,b) assigned a block.
- Inside each block, examine many different orbits: one thread per orbit.
  $\implies$ We check $512 - 1024$ different orbits for each (a, b).

Parallelisation of search for sinks on Haddock cluster

- Haddock
  `http://www.uppmax.uu.se/the-haddock-cluster`
  28 compute nodes with dual Tesla M2075 GPUs.

- Each parameter value (a,b) assigned a block.
- Inside each block, examine many different orbits: one thread per orbit.
  $\Longrightarrow$ We check $512 - 1024$ different orbits for each (a, b).
- $h^n_{a,b}$ iterations are computed in: double, double-double, triple-double, etc;
  $n \sim 10^6 - 10^9$.

Parallelisation of search for sinks on Haddock cluster

- Haddock
  `http://www.uppmax.uu.se/the-haddock-cluster`
  28 compute nodes with dual Tesla M2075 GPUs.

- Each parameter value (a,b) assigned a block.
- Inside each block, examine many different orbits: one thread per orbit.
  $\Longrightarrow$ We check $512 - 1024$ different orbits for each (a, b).
- $h^n_{a,b}$ iterations are computed in: double, double-double, triple-double, etc;
  $n \sim 10^6 - 10^9$.
- Code example:

```
#define prec 4   /*kernel to be run using quad prec*/
__device__ void henon_iterate(double x0, double y0,
                  double a, double b, long int ITER) {
    /*init multi_prec template vars*/
    multi_prec<prec,double> x_i(x0);
    multi_prec<prec,double> y_i(y0);
    multi_prec<prec,double> x_old;
    for (long int i=1; i <= ITER; i++) {
        /*Compute iterates*/
        x_i = y_i + 1.0 - a*x_i*x_i;
        y_i = b*x_old;
    }
}
```

Found Hénon sinks [Galias & Tucker, 2013]

Closest to classical parameters, so far: $a = 1.399999486944, b = 0.3$.

a	p	d	r	λ_1
1.399922051	25	$5.522-12$	$2.473-12$	-0.00132
1.39997174948	30	$1.354-11$	$3.561-12$	-0.01887
1.3999769102	18	$3.207-09$	$1.014-09$	-0.05306
1.39998083519	24	$1.703-11$	$7.384-12$	-0.02819
1.399984477	20	$8.875-10$	$4.076-10$	-0.05099
1.39999492185	22	$3.686-11$	$1.531-11$	-0.09600
1.3999964733062	39	$2.784-13$	$1.115-13$	-0.03547
1.399999486944	33	$1.110-12$	$6.901-13$	-0.01843
1.40000929916	25	$1.118-11$	$5.128-12$	-0.08379
1.4000227433	21	$2.262-10$	$7.901-11$	-0.05612
1.40002931695	27	$5.782-11$	$2.646-11$	-0.01140
1.40006377472	27	$8.692-11$	$3.810-11$	-0.05636
1.40006667358	24	$6.278-11$	$2.646-11$	-0.01112
1.4000843045	27	$9.400-11$	$4.572-11$	-0.06870
1.40009110518	22	$3.493-11$	$1.531-11$	-0.02157
1.4000967515	26	$2.463-10$	$1.365-10$	-0.13233

Conclusion

- We wanted a feasibility study for multiprec algorithms on GPU;

Conclusion

- We wanted a feasibility study for multiprec algorithms on GPU;
- Careful implementation: avoid dynamic memory allocation on the GPU; use templates; use in-place algorithms to spare local memory; avoid memory spill/loads.

Conclusion

- We wanted a feasibility study for multiprec algorithms on GPU;
- Careful implementation: avoid dynamic memory allocation on the GPU; use templates; use in-place algorithms to spare local memory; avoid memory spill/loads.
- Performance: peak number of Hénon map orbits/second for double vs. extended precision on Tesla GPU[C2075] using 10^6 iterations/orbit

Precision	Perf
double	102398
2 doubles	1798
3 doubles	515
4 doubles	180
5 doubles	98

Conclusion

- We wanted a feasibility study for multiprec algorithms on GPU;
- Careful implementation: avoid dynamic memory allocation on the GPU; use templates; use in-place algorithms to spare local memory; avoid memory spill/loads.
- Performance: peak number of Hénon map orbits/second for double vs. extended precision on Tesla GPU[C2075] using 10^6 iterations/orbit

Precision	Perf
double	102398
2 doubles	1798
3 doubles	515
4 doubles	180
5 doubles	98

- Correct rounding for operations with FP expansions ?

Conclusion

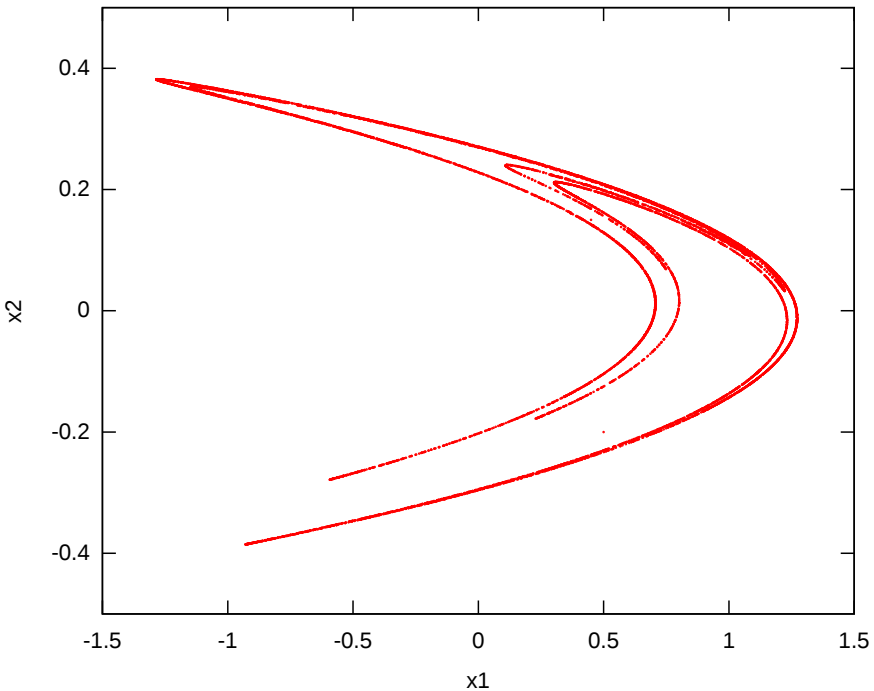
- We wanted a feasibility study for multiprec algorithms on GPU;
- Careful implementation: avoid dynamic memory allocation on the GPU; use templates; use in-place algorithms to spare local memory; avoid memory spill/loads.
- Performance: peak number of Hénon map orbits/second for double vs. extended precision on Tesla GPU[C2075] using 10^6 iterations/orbit

Precision	**Perf**
double	102398
2 doubles	1798
3 doubles	515
4 doubles	180
5 doubles	98

- Correct rounding for operations with FP expansions ?
- Iterate other dynamical systems ?

Conclusion

Strange Attractors

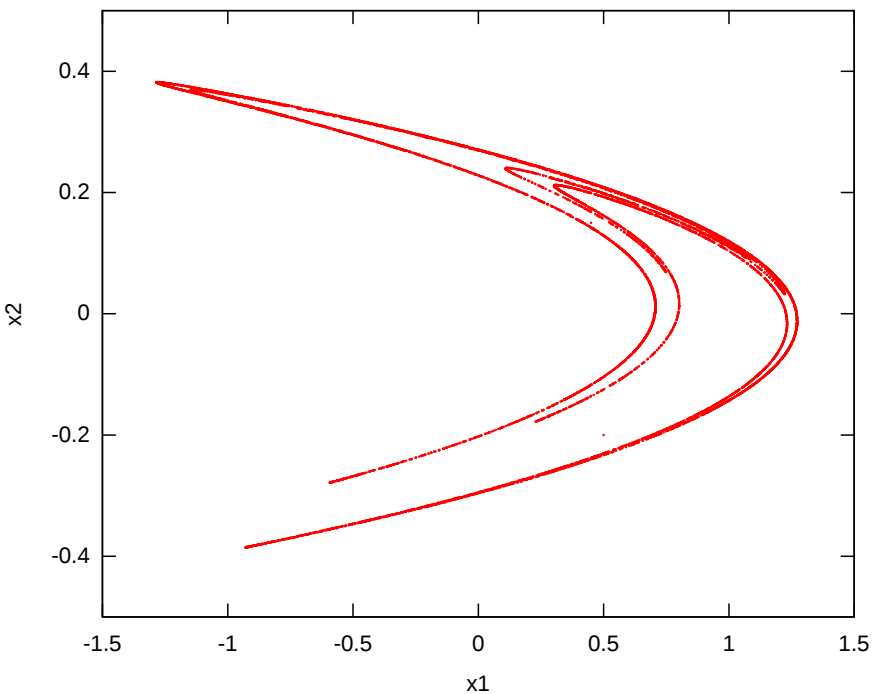


Quoting Ruelle...

I asked Floris Takens if he had created this remarkably successful expression. Here is his answer: "Did you ever ask God whether he created this damned Universe? ... I don't remember anything ... I often create without remembering it..." The creation of strange attractors thus seems to be surrounded by clouds and thunder. Anyway, the name is beautiful, and well suited to these astonishing objects, of which we understand so little.

Conclusion

Strange Attractors



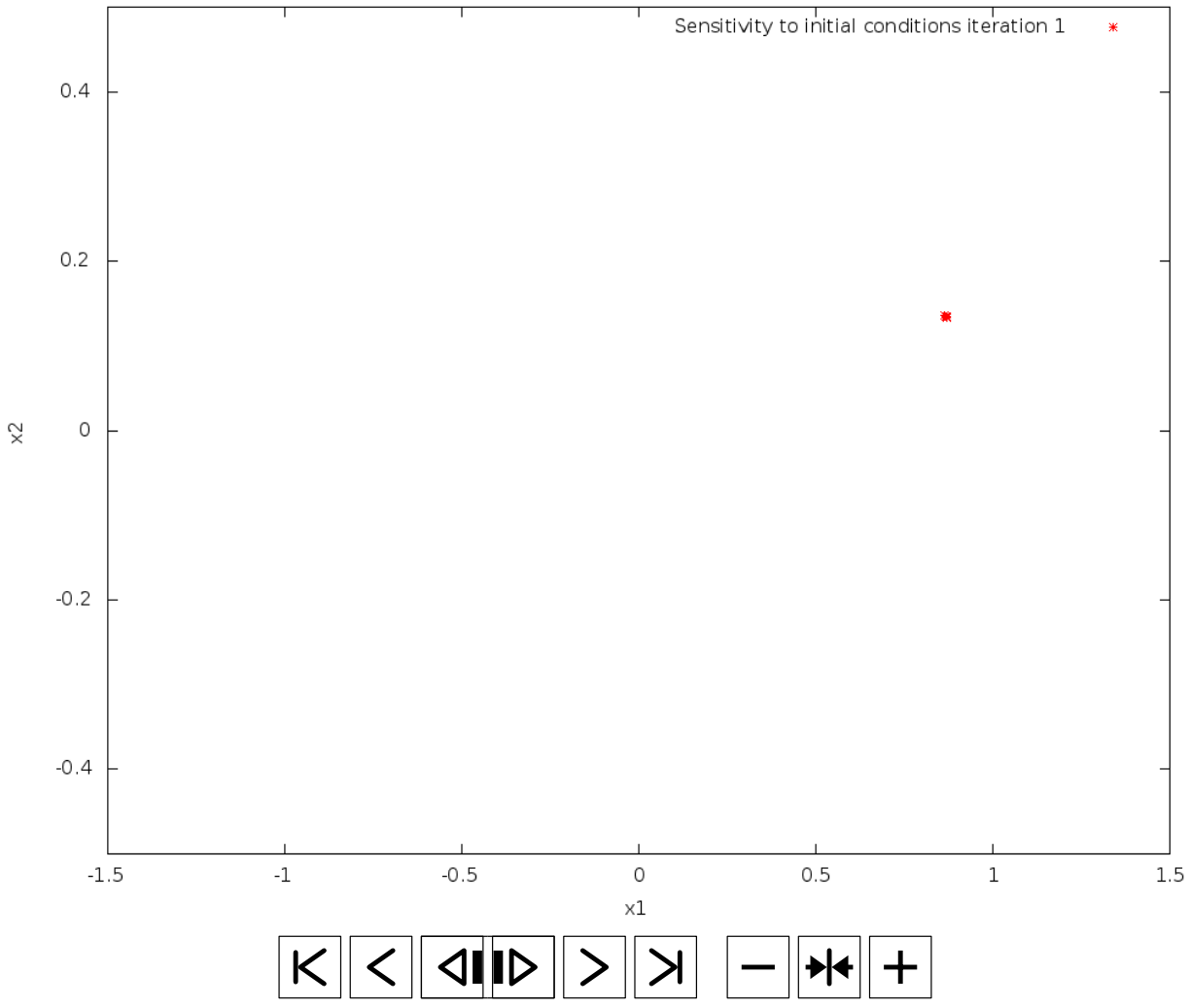
Quoting Ruelle...

I asked Floris Takens if he had created this remarkably successful expression. Here is his answer: "Did you ever ask God whether he created this damned Universe? ... I don't remember anything ... I often create without remembering it..." The creation of strange attractors thus seems to be surrounded by clouds and thunder. Anyway, the name is beautiful, and well suited to these astonishing objects, of which we understand so little.

Hénon Attractor - Sensitivity to initial conditions

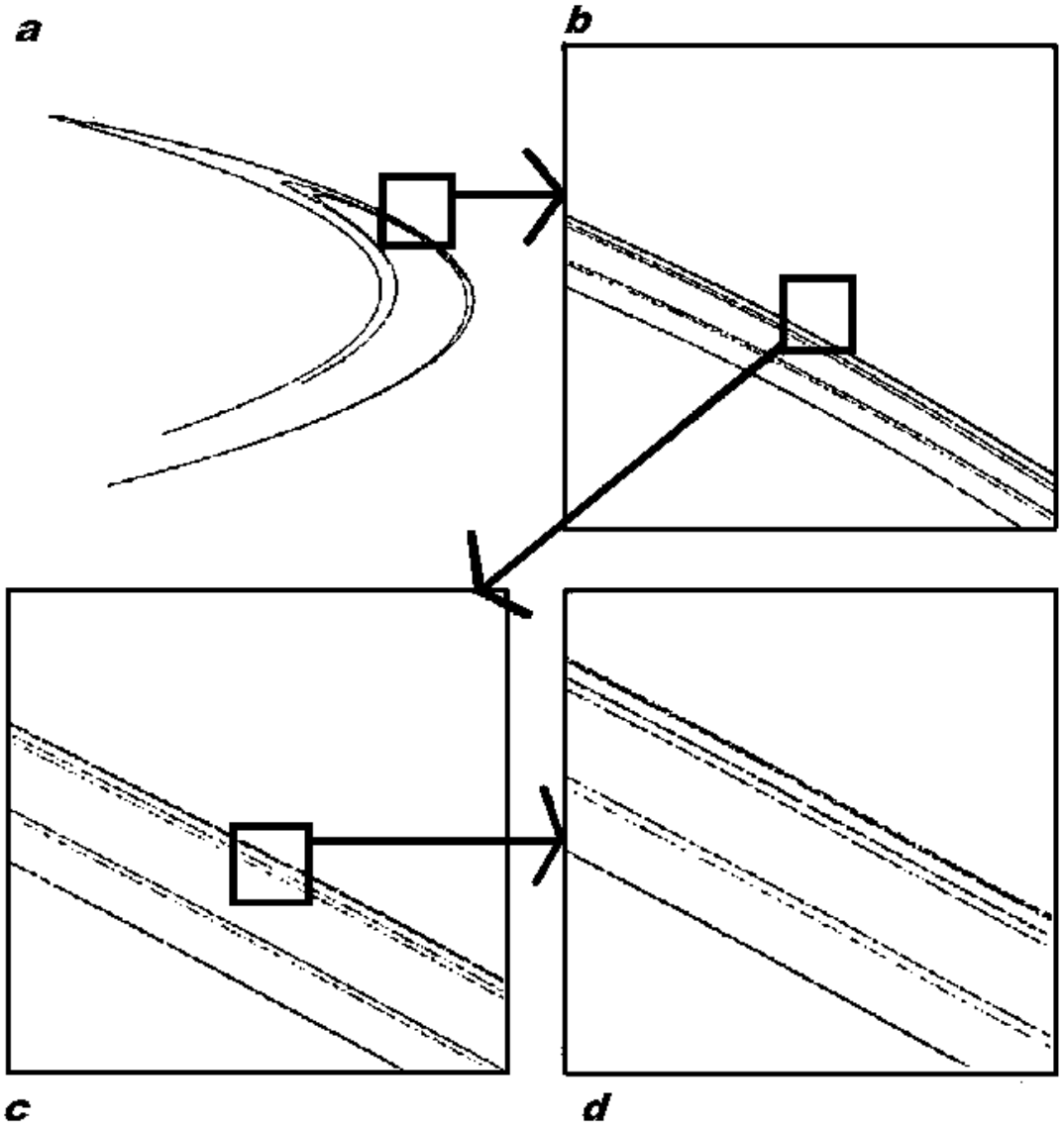
Hénon Map

$$h_{a,b}(x_1, x_2) = (1 + x_2 - ax_1^2, bx_1)$$



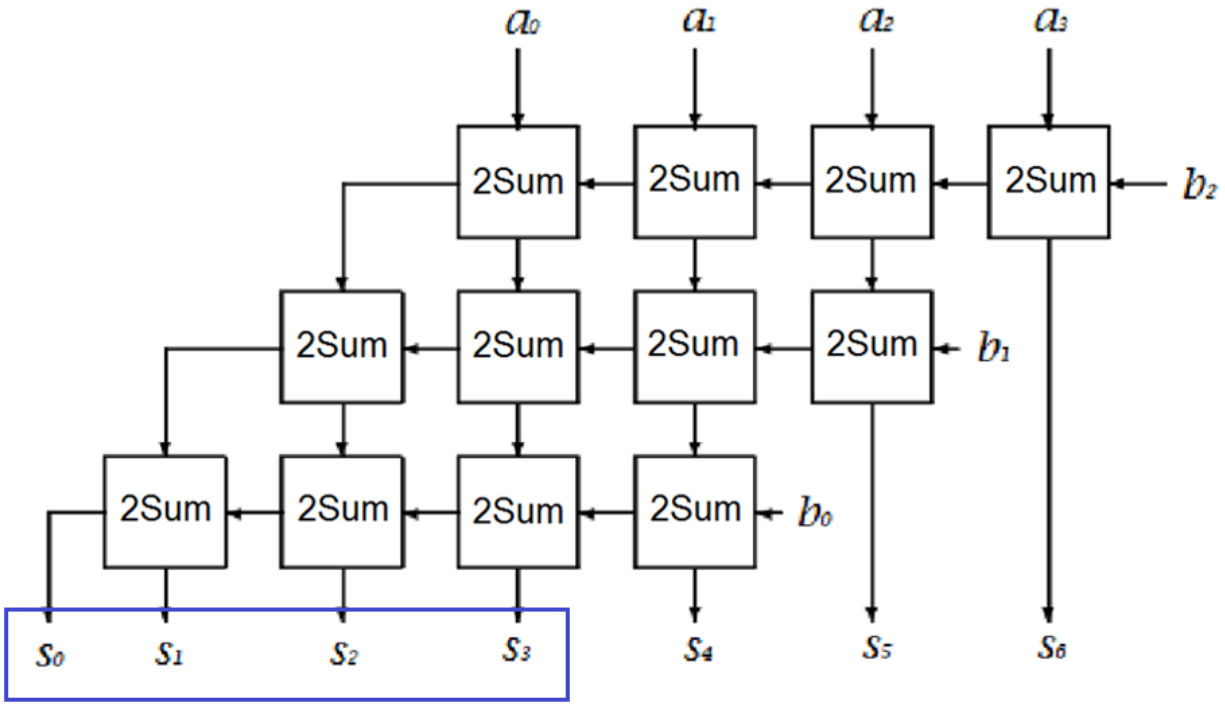
Back

Hénon Attractor - Fractal Dimension

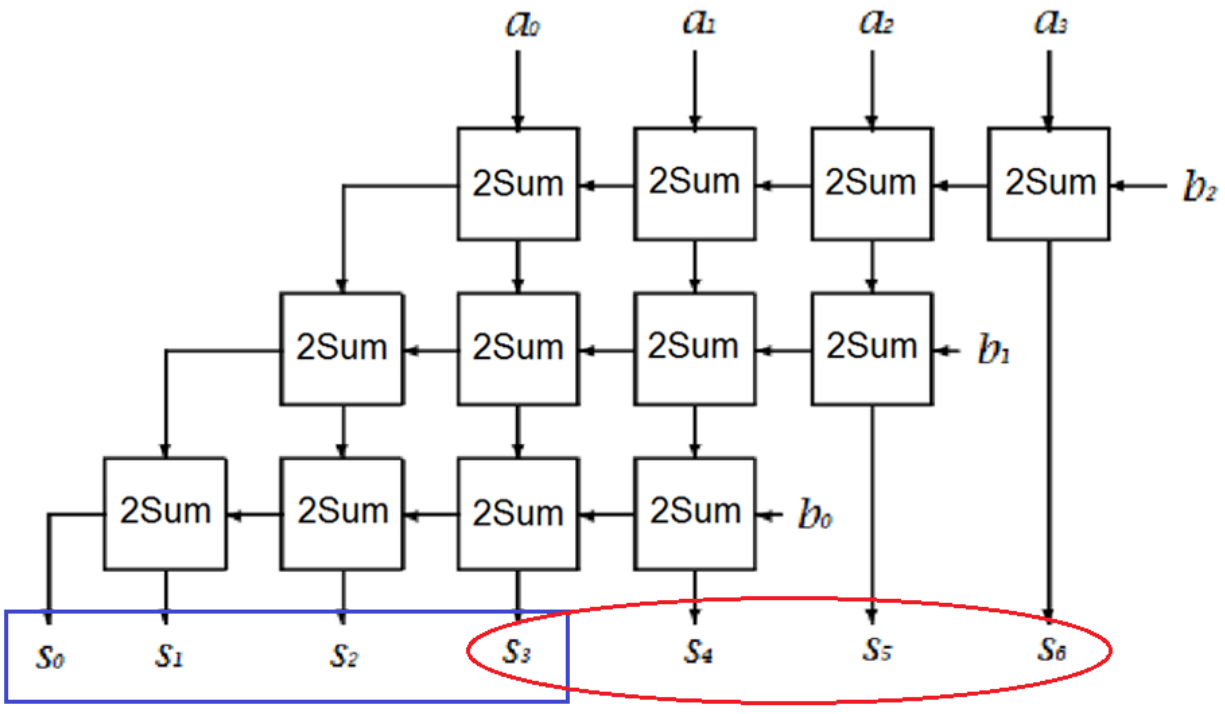


Back

Shewchuck's ExpansionSum



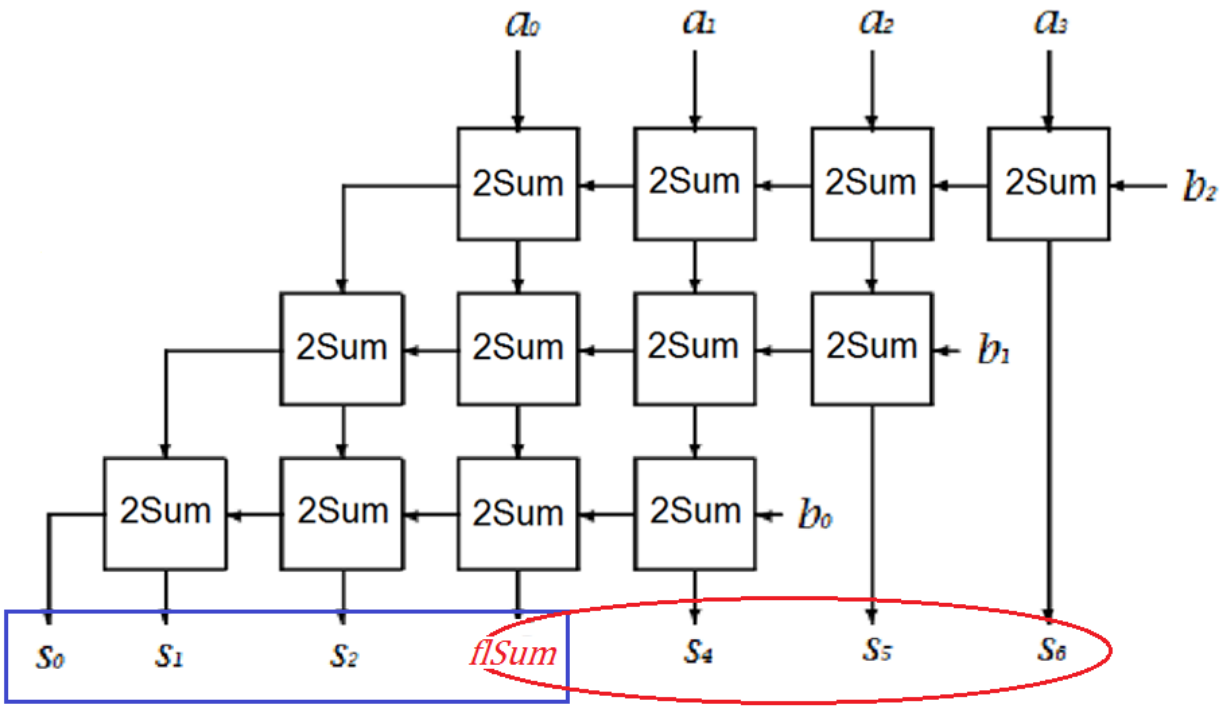
Shewchuck's ExpansionSum



$$u = \mathsf{RU}\left(\sum_{i=r-1}^{n+m-1} s_i\right); \qquad flSum = \mathsf{RN}\left((u+d)/2\right);$$

$$d = \mathsf{RD}\left(\sum_{i=r-1}^{n+m-1} s_i\right); \qquad \varepsilon = \max\left(\mathsf{RU}\left(flSum - d\right), \mathsf{RU}\left(u - flSum\right)\right);$$

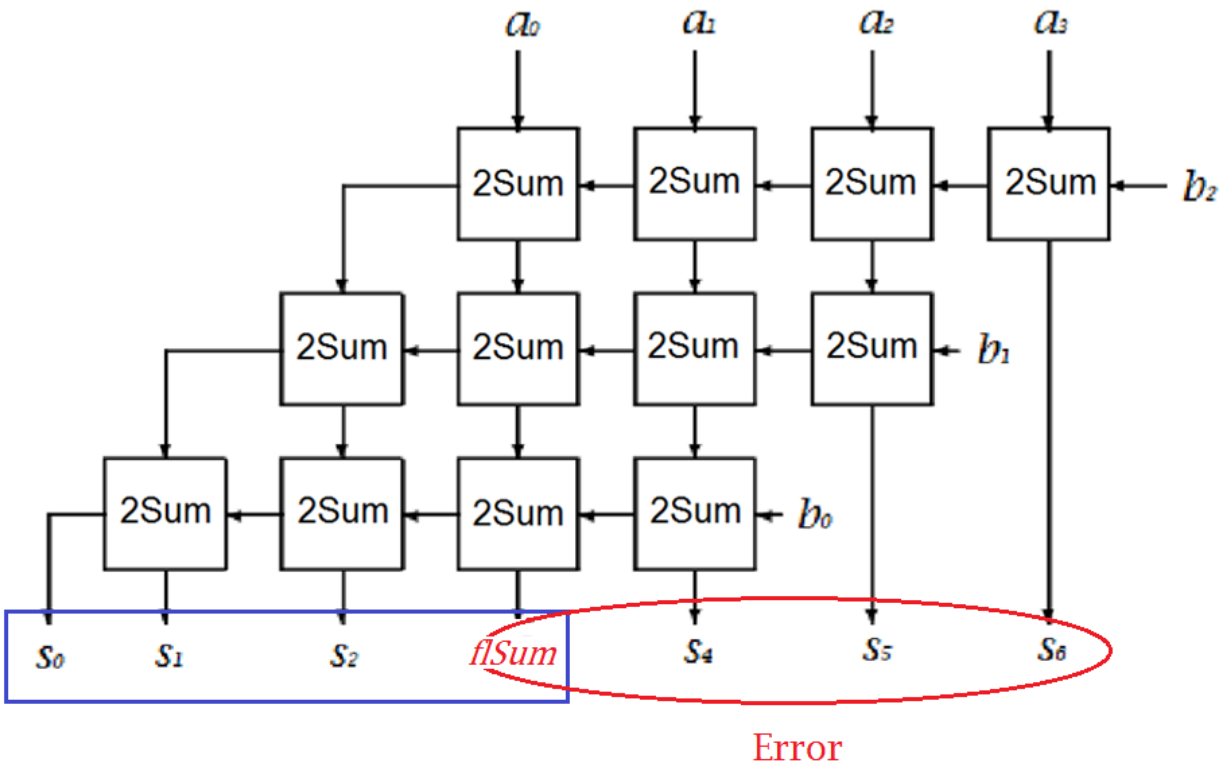
Shewchuck's ExpansionSum



$$u = \mathsf{RU}\left(\sum_{i=r-1}^{n+m-1} s_i\right); \qquad flSum = \mathsf{RN}\left((u+d)/2\right);$$

$$d = \mathsf{RD}\left(\sum_{i=r-1}^{n+m-1} s_i\right); \qquad \varepsilon = \max\left(\mathsf{RU}\left(flSum - d\right), \mathsf{RU}\left(u - flSum\right)\right);$$

Shewchuck's ExpansionSum



$$u = \mathsf{RU}\left(\sum_{i=r-1}^{n+m-1} s_i\right); \qquad flSum = \mathsf{RN}\left((u+d)/2\right);$$

$$d = \mathsf{RD}\left(\sum_{i=r-1}^{n+m-1} s_i\right); \qquad \varepsilon = \max\left(\mathsf{RU}\left(flSum - d\right), \mathsf{RU}\left(u - flSum\right)\right);$$

Interval arithmetic support: $(\sum_{i=1}^{r} s_i, \varepsilon)$ with very little overhead.