Um Estudo da Dinâmica do Mapa Quadrático

Eduardo Tsuyoshi Ochiai

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA OBTENÇÃO DO GRÁU DE
MESTRE EM
MATEMÁTICA APLICADA

Orientador : Cláudio Possani

Suporte financeiro do CNPq

São Paulo, 29 de Junho de 1994

UM ESTUDO DA DINÂMICA DO MAPA QUADRÁTICO

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e apresentada por Eduardo Tsuyoshi Ochiai e aprovada
pela Comissão Julgadora

São Paulo, 06 de

Banca Examinadora:

- Prof. Dr. Cláudio Possani (orientador) - IME USP
- Prof. Dr. Edson Vargas - IME USP
- Prof. Dr. Robert Roussarie - Univ. Dijon - França

Agradecimentos

Aos meus pais.

Ao meu orientador, Cláudio Possani.

Aos profs. Jorge Beloqui e Zara Abud, por todo o auxílio desde o início de meu mestrado no IME.

Aos profs. Antônio L. Pereira, Edson de Faria e Edson Vargas, pelas sugestões e observações, que foram importantes para a conclusão deste trabalho.

Ao meu amigo Célio, pela grande ajuda na digitação da tese em TEX, e seu apoio constante; ao meu amigo Henrique, também por seu apoio; e aos inúmeros colegas de pós-graduação do IME e do Inst. Física, pelas conversas, discussões e sugestões. Em especial, a Luiz Donizetti, por me introduzir ao mundo dos softwares matemáticos, e me ajudar a desvendar a linguagem C. Por fim, a Fernando Barbi, por ter gentilmente cedido sua cópia do software *Mathematica* para uso na elaboração da dissertação.

Ao Prof. Jacques C.Bouchara(*In memorian*)

INTRODUÇÃO

A dinâmica unidimensional apresenta a maioria dos fenômenos observados na teoria de sistemas dinâmicos, como pontos periódicos, órbitas densas e dependência sensível nas condições iniciais. Nosso objetivo aqui será estudar alguns resultados sobre a dinâmica unidimensional, em particular para o mapa quadrático $F_\mu(x) = \mu \cdot x \cdot (1 - x)$, para $\mu > 4$.

Inicialmente, no capítulo 1, introduzimos algumas definições básicas para o estudo de mapas na reta real, incluindo a dimensão de Hausdorff e a entropia, ilustrando com alguns exemplos, e terminando com a definição de mapa caótico.

No capítulo 2, as definições e resultados do cap. 1 são aplicados ao mapa quadrático. É introduzida a dinâmica simbólica, e é apresentado o mapa conhecido como "shift", como um exemplo de conjugação topológica. Utilizando esse mapa, é obtida a entropia do mapa quadrático (usando o resultado de que a entropia é invariante por conjugação topológica), e é demonstrado que o mapa quadrático, na faixa dada do parâmetro μ , é caótico.

O capítulo 3 é dedicado à obtenção de alguns resultados numéricos: o cálculo dos pontos periódicos do mapa quadrático ($\mu > 4$) e das órbitas densas no conjunto invariante do mesmo. É apresentada uma demonstração alternativa, sem o uso da dinâmica simbólica, de que o mapa quadrático é caótico.

No apêndice A, são utilizadas algumas idéias desenvolvidas no cap. 3 para tentar obter estimativas de limites superior e inferior da dimensão de Hausdorff do conjunto invariante Λ_μ do mapa quadrático. No entanto, essas estimativas só são válidas se a medida de Hausdorff de Λ_μ (na sua própria dimensão) é não-nula, o que não demonstramos; assim, esses resultados ficam abertos.

O apêndice B mostra algumas outras funções na reta para as quais talvez pudessem ser aplicados métodos como os desenvolvidos no capítulo 3. No entanto, as demonstrações das proposições do cap. 3 são específicas para o mapa quadrático, as-

- sim teriam de ser modificadas para abranger outras funções. A título de ilustração são calculados alguns pontos(usando esses métodos), para cada mapa, que se comportam, dentro da precisão usada, como pontos periódicos.

No apêndice C, é construído um mapa bidimensional que se comporta como o "horseshoe " em seu conjunto invariante. Usando métodos semelhantes aos usados no cap. 3, pode-se calcular pontos periódicos para esse mapa, e assim obter alguma informação indireta sobre o "horseshoe ".

O apêndice D é uma listagem de resultados numéricos, referentes ao cap. 3.

Resumo

Alguns conceitos básicos da dinâmica de mapas unidimensionais, como órbitas, pontos fixos e periódicos, conjuntos invariantes e conjugação topológica, são apresentados e discutidos. Esses conceitos, assim como a definição de um mapa caótico, são aplicados ao mapa quadrático $\mu \cdot x \cdot (1-x)$, para μ maior que 4. A dinâmica simbólica é introduzida para obter um mapa, o shift, que é topologicamente conjugado ao mapa quadrático sobre seu conjunto invariante, mas cuja análise é muito mais simples, e pode ser estendida naturalmente ao próprio mapa quadrático. Alguns resultados numéricos são obtidos, inclusive o cálculo dos pontos periódicos e órbitas densas do mapa quadrático. Os conceitos de dimensão de Hausdorff de um conjunto e da entropia de um mapa são também apresentados e aplicados ao mapa quadrático. A entropia desse mapa é calculada (via dinâmica simbólica), e é sugerido um método para estimar limitações superior e inferior para a dimensão de Hausdorff do seu conjunto invariante.

Abstract

Some basic concepts on the dynamics of real one-dimensional maps, such as orbits, fixed and periodic points, invariant sets and topological conjugacy, are presented and discussed. Those concepts, along with the definition of a chaotic map, are applied to the quadratic map $\mu \cdot x \cdot (1-x)$, for μ greater than 4. Symbolic dynamics is introduced to obtain a map, the shift, which is topologically conjugate to the quadratic map on its invariant set but yields a much easier analysis, which then naturally extends to the quadratic map itself. Some numerical results are obtained, including the computation of the periodic points and dense orbits of the quadratic map. The concepts of Hausdorff dimension of a set and entropy of a map are also presented and applied to the quadratic map. The entropy of that map is calculated (via symbolic dynamics), and a method is suggested to estimate upper and lower bounds for the Hausdorff dimension of its invariant set.

Índice

1	Introdução	3
1.1	Definições básicas	3
1.2	Caos	8
1.3	Dimensão de Hausdorff	9
1.4	Entropia	11
2	Dinâmica do mapa quadrático	13
2.1	Conjunto invariante	14
2.2	Dinâmica simbólica	16
2.3	Entropia	20
3	Mapa quadrático: análise numérica	22
3.1	Pontos periódicos	22
3.2	Órbitas densas em Λ_μ	28
3.3	Caos	29
3.4	Comparação com a dinâmica simbólica	31
3.5	Resultados numéricos	32
A	Observações sobre $\text{HD}(\Lambda_\mu)$	33
B	Outras funções	36
C	Um exemplo bidimensional	41
D	Resultados numéricos	44
	Bibliografia	49

CAPÍTULO 1

Introdução

A dinâmica unidimensional apresenta a maioria dos fenômenos estudados na teoria de sistemas dinâmicos, como órbitas periódicas, conjuntos invariantes e bifurcações. Estudaremos aqui a dinâmica de mapas unidimensionais, ou seja, funções na reta real, a valores reais. O objetivo básico é estudar sequências $\{x, f(x), f^2(x), \dots\}$, ou seja, o comportamento de pontos x da reta real por iteração de uma função f .

1.1 Definições básicas

Definição 1.1.1 A órbita direta de $x \in R$ pela função f é o conjunto de pontos

$$O_f^+(x) = \{x, f(x), f^2(x), f^3(x), \dots\}$$

Se a função f é um homeomorfismo, então a órbita inversa de $x \in R$ pela função f é:

$$O_f^-(x) = \{x, f^{-1}(x), f^{-2}(x), f^{-3}(x), \dots\}$$

Definição 1.1.2 Um ponto $x \in R$ é ponto fixo de f se $f(x) = x$. Um ponto $x \in R$ é ponto periódico de f se $\exists n \in N^*$ tal que $f^n(x) = x, n > 0$, e n é o período de f .

Definição 1.1.3 Um ponto $x \in R$ é eventualmente periódico de período n se existe $m \in N^*$ tal que $f^m(x)$ seja ponto periódico de período n .

Definição 1.1.4 *Seja p um ponto periódico de período n . Um ponto x é diretamente assintótico a p se $\lim_{i \rightarrow \infty} f^{in}(x) = p, i \in N$. O conjunto estável de p , $W^s(p)$, é o conjunto dos pontos diretamente assintóticos a p .*

Definição 1.1.5 *Se f é homeomorfismo, então x é um ponto inversamente assintótico a p se $\lim_{i \rightarrow \infty} f^{-in}(x) = p, i \in N$. O conjunto instável de p , $W^u(p)$, é o conjunto dos pontos inversamente assintóticos a p .*

Exemplo 1.1.6 A função $f(x) = x^2 - 1$.

A função f tem os pontos fixos $\{\frac{(1-\sqrt{5})}{2}, \frac{(1+\sqrt{5})}{2}\}$, e a órbita periódica $\{-1, 0\}$.

O ponto $x = 1$ é eventualmente periódico, pois $f(1) = 0$.

Podemos usar a seguinte construção gráfica para analisar órbitas de pontos por f :

- I. Dado x , marcar o ponto (x, x) na bisetritz B ;
- II. Traçar uma reta vertical por (x, x) e marcar o ponto em que ela cruza o gráfico de f , ou seja, o ponto $(x, f(x))$;
- III. Traçar uma reta horizontal por $(x, f(x))$ e marcar o ponto em que ela cruza a bisetritz B , ou seja, o ponto $(f(x), f(x))$;
- IV. Repetir II, agora com o ponto $(f(x), f(x))$.

Por este processo, vemos que os pontos no intervalo $(1, \frac{(1+\sqrt{5})}{2})$ são assintóticos ao ponto fixo $\frac{(1+\sqrt{5})}{2}$.

Definição 1.1.7 Um ponto fixo p de uma função $f : I \rightarrow R$ é atrator se existe um intervalo aberto limitado $A \subset I$ tal que $p \in I$ e, para qualquer x em A , $\lim_{n \rightarrow \infty} f^n(x) = p$.

Definição 1.1.8 Um ponto fixo p de f é repulsor se existe um intervalo aberto limitado $A \subset I$ tal que $p \in I$ e, para qualquer x em A , existe $n > 0$ tal que $f^n(x) \notin A$.

Definição 1.1.9 Seja a função $f : I \rightarrow R$, de classe C^1 . Uma condição suficiente para que um ponto fixo $p \in I$ seja atrator é que $|f'(p)| \leq k < 1$, e uma condição suficiente para que p seja repulsor é que $|f'(p)| \geq k > 1$.

Prova. Seja $|f'(p)| \leq k < 1$. Como f é de classe C^1 , existe uma vizinhança $A \subset I$ de p tal que, para qualquer x em A , $|f'(x)| \leq k$. Seja então um ponto qualquer $x \in A$; pelo teorema do valor médio, temos:

$$|f(x) - f(p)| = |f(x) - p| \leq k |x - p| < |x - p|$$

Como $|f(x) - p| < |x - p|$, o ponto $f(x)$ também está em A ; logo, podemos aplicar novamente o teorema do valor médio, obtendo

$$|f^2(x) - f(p)| = |f^2(x) - p| \leq k |f(x) - p| \leq k^2 |x - p|$$

Continuando o processo, obtemos

$$|f^n(x) - p| \leq k^n |x - p|$$

Logo, $\lim_{n \rightarrow \infty} f^n(x) = p$. $\square$

Seja, agora, $|f'(p)| \geq k > 1$. Como antes, existe $A \subset I$ tal que, para qualquer x em A , $|f'(x)| \geq k$. Seja então $x \in A$. Novamente pelo teorema do valor médio,

$$|f(x) - f(p)| = |f(x) - p| \geq k |x - p| > |x - p|$$

Continuando o processo,

$$|f^n(x) - p| \geq k^n |x - p|$$

Logo, existe n tal que $|f^n(x) - p| > d(A)$, o que implica que $f^n(x) \notin A$. $\square$

Exemplo 1.1.10 A função $f(x) = \sqrt{x}$, para $x \geq 0$.

A função f tem os pontos fixos $\{0, 1\}$.

A derivada de f em $x = 1$ é $f'(1) = \frac{1}{2}$. Logo, pela proposição 1.1.9, o ponto $x = 1$ é atrator. Isso pode ser facilmente observado numa calculadora, iniciando com qualquer valor positivo, e aplicando a raiz quadrada sucessivamente.

Definição 1.1.11 Um conjunto $S \subset \mathbb{R}$ é invariante pela função f se $f(S) = S$.

Exemplo 1.1.12 A função $f(x) = x^3$.

Seja o intervalo $S = [-1, 1]$. Os pontos $x = \pm 1$ são fixos, e para qualquer ponto $x \in (-1, 1)$, temos $|x| < 1$, logo (v. figura abaixo)

$$|f(x)| = |x|^3 < |x|$$

ou seja, todos os pontos de $(-1, 1)$ são assintóticos à origem $x = 0$. Segue que

$$f(S) = S.$$

Definição 1.1.13 *Sejam duas funções $f : U \rightarrow U$ e $g : V \rightarrow V$. Uma função $h : U \rightarrow V$ é uma conjugação topológica entre f e g se h é um homeomorfismo e $h \circ f = g \circ h$. Graficamente,*

$$\begin{array}{ccc} U & \xrightarrow{f} & U \\ h \downarrow & & \downarrow h \\ V & \xrightarrow{g} & V \end{array}$$

A existência de uma conjugação topológica entre dois mapas mostra que a dinâmica deles é essencialmente a mesma; precisamente, há uma relação biunívoca entre os pontos fixos e periódicos dos dois mapas.

Proposição 1.1.14 *Sejam duas funções $f : U \rightarrow U$ e $g : V \rightarrow V$, e $h : U \rightarrow V$ uma conjugação topológica entre f e g . Então, (a) se p é ponto fixo de f , $h(p)$ é ponto fixo de g ; (b) se q é ponto periódico de período n de f , $h(q)$ é ponto periódico de período n de g .*

Prova. (a) Seja $f(p) = p$. Pela definição de conjugação,

$$h(f(p)) = g(h(p)) \Rightarrow h(p) = g(h(p))$$

Ou seja, $h(p)$ é ponto fixo de g .

(b) Seja $f^n(q) = q$. Lembrando que h é homeomorfismo, portanto é inversível,

$$g = h \circ f \circ h^{-1}$$

Compondo n vezes,

$$g^n = (h \circ f \circ h^{-1})^n = h \circ f \circ h^{-1} \circ h \circ f \circ h^{-1} \circ \dots \circ h \circ f \circ h^{-1}$$

Observe-se que, como h é homeomorfismo, $h^{-1} \circ h = \text{Identidade}$; logo,

$$g^n = h \circ f^n \circ h^{-1}$$

Aplicando em $h(q)$,

$$g^n(h(q)) = h(f^n(h^{-1}(h(q)))) = h(f^n(q)) = h(q)$$

$\Rightarrow h(q)$ é ponto periódico de período n de g . $\square$

1.2 Caos

Um sistema dinâmico caótico é, intuitivamente, um sistema cuja evolução temporal a partir de dadas condições iniciais mude consideravelmente se essas condições iniciais forem alteradas infinitesimalmente, tal que as órbitas "se misturem" e que tenha, ainda assim, "muitas" órbitas periódicas. Existem várias definições de caos; aqui seguimos [1].

Definição 1.2.1 *Uma função $f : S \rightarrow S$ é topologicamente transitiva se, dados quaisquer abertos $U, V \subset S$, existe $k > 0$ tal que $f^k(U) \cap V \neq \emptyset$. Uma condição suficiente para que f seja topologicamente transitiva é que exista uma órbita de f densa em S .*

Definição 1.2.2 *Uma função $f : I \rightarrow I$ tem dependência sensitiva nas condições iniciais se existe $\delta > 0$ tal que, dados $x \in I$ e V uma vizinhança de x , existe $y \in V$ tal que*

$$|f^n(x) - f^n(y)| > \delta$$

para algum $n > 0$.

Definição 1.2.3 *Seja $f : S \rightarrow S$. A função f é caótica em S se as seguintes condições são satisfeitas:*

1. *f é topologicamente transitiva ;*
2. *f tem dependência sensitiva nas condições iniciais;*
3. *O conjunto dos pontos periódicos de f é denso em S .*

Esta definição será usada nos capítulos 2 e 3 para analisar o comportamento do mapa quadrático.

1.3 Dimensão de Hausdorff

Para medir conjuntos de R^n que não sejam "suaves", como por exemplo a curva de Koch ou um conjunto de Cantor, é necessária uma definição mais geral de dimensão, que leve em conta a complexidade do "contorno" de tais conjuntos. No nosso caso, o problema é obter um número que diferencie dois conjuntos de medida (de Lebesgue) nula na reta; por exemplo, o conjunto de Cantor e um conjunto finito de pontos. Para isso, usamos a dimensão de Hausdorff, que é definida por meio de um processo-limite de coberturas do conjunto por abertos (ou fechados) de R^n .

Definição 1.3.1 *Seja D um subconjunto de R^n . Uma família de subconjuntos $\{A_i\}_{i=1,2,3,\dots}$ de R^n é um δ -recobrimento de D se*

$$D \subset \bigcup_{i=1}^{\infty} A_i$$

e

$$d(A_i) \leq \delta, \forall i \in N^*.$$

onde $d(A_i)$ é o diâmetro do conjunto A_i .

Sejam D um subconjunto de R^n e $s \geq 0$ um número real. Então, define-se

$$H_{\delta}^s(D) = \inf \left\{ \sum_{i=1}^{\infty} d(A_i)^s \mid \{A_i\} \text{ é um } \delta\text{-recobrimento de } D \right\}$$

Se $\{B_i\}$ é um ϵ -recobrimento de D , $\epsilon < \delta$, então $\{B_i\}$ é também um δ -recobrimento de D . Logo,

$$H_{\epsilon}^s(D) \geq H_{\delta}^s(D)$$

Definição 1.3.2 *Define-se*

$$H^s(D) = \sup_{\delta \geq 0} H_{\delta}^s(D) = \lim_{\delta \rightarrow 0} H_{\delta}^s(D)$$

como a medida s -dimensional de Hausdorff de D . (ver [3]).

Definição 1.3.3 *Usando a def. 1.3.2, demonstra-se que existe um número real s tal que*

$$H^t(D) = 0, \quad t > s$$

$$H^t(D) = \infty, \quad t < s$$

s é chamado dimensão de Hausdorff de D , e denota-se

$$s = HD(D).$$

Para verificar que existe esse número, observamos que, se $s < t$, e $\{A_i\}$ é um δ -recobrimento de E ,

$$\sum_{i=1}^{\infty} d(A_i)^s = \sum_{i=1}^{\infty} d(A_i)^{s-t} \cdot d(A_i)^t \geq \delta^{s-t} \cdot \sum_{i=1}^{\infty} d(A_i)^t$$

Logo, pela definição da medida de Hausdorff, vem :

$$H^s(E) \geq \delta^{s-t} \cdot H^t(E) = \frac{1}{\delta^{t-s}} \cdot H^t(E)$$

(onde $t - s > 0$).

Fazendo $\delta \rightarrow 0$ segue que se $H^t(E)$ é finito, $H^s(E)$ é infinito, e se $H^s(E)$ é finito, $H^t(E)$ é zero. Concluimos que existe o número s definido acima. $\square$

Lema 1.3.4 *Seja uma função $f : A \subset R^n \rightarrow R^n$, tal que $\exists k > 0$ para o qual*

$$|f(x) - f(y)| \leq k |x - y|, \quad \forall x, y \in A,$$

então, se $E \subset A$, e $0 < H^s(E) < \infty$, onde $s = HD(E)$, verifica-se

$$H^s(f(E)) \leq k^s H^s(E).$$

Prova . Seja $\{A_i\}$ um δ -recobrimento de E . Então, como $d(f(A_i)) \leq k \{d(A_i \cap E)\}$, $\{f(A_i)\}$ é um $k \cdot \delta$ -recobrimento de E , e

$$\sum_{i=1}^{\infty} d(f(A_i))^s \leq k^s \cdot \sum_{i=1}^{\infty} d(A_i)^s$$

Logo,

$$H_{k \cdot \delta}^s(f(E)) \leq k^s \cdot H_{\delta}^s(E).$$

Tomando o limite $\delta \rightarrow 0$, vem:

$$H^s(f(E)) \leq k^s \cdot H^s(E). \quad \square$$

Lema 1.3.5 *Seja uma função $f : A \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$, tal que $\exists k > 0$ para o qual*

$$|f(x) - f(y)| \geq k|x - y|, \quad \forall x, y \in A,$$

então, se $E \subset A$ e $0 < H^s(E) < \infty$, onde $s = HD(E)$, verifica-se

$$H^s(f(E)) \geq k^s H^s(E).$$

Prova. Análoga à anterior.

1.4 Entropia

Definimos agora um número característico da dinâmica de um mapa da reta $f : I \rightarrow I$, a *entropia topológica*. Ela mede a complexidade das órbitas de pontos de I pelo mapa. Seguimos para isso a definição de [5].

Definição 1.4.1 *Seja $K \subset \mathbb{R}$ um compacto, e $f : K \rightarrow K$ uma função contínua. Então, um conjunto $X \subset K$ é (n, ϵ) -separado se, dados quaisquer pontos $x, y \in X$, existe um inteiro $0 \leq j < n$ tal que $d(f^j(x), f^j(y)) > \epsilon$. Em outras palavras, um conjunto $X \subset K$ é (n, ϵ) -separado se, dentro de uma "tolerância" ϵ , as órbitas (até a n -ésima iteração) de pontos de X são distintas.*

Definição 1.4.2 *Seja $s_n(\epsilon, K)$ definido como a máxima cardinalidade de conjuntos (n, ϵ) -separados contidos em K . Então, definimos:*

$$s(\epsilon, K) = \limsup_{n \rightarrow \infty} \frac{1}{n} \log s_n(\epsilon, K).$$

Observando que se $\epsilon_1 < \epsilon_2$ um conjunto (n, ϵ_2) -separado é também (n, ϵ_1) -separado, vem:

$$s(\epsilon_1, K) \geq s(\epsilon_2, K).$$

Definição 1.4.3 *Seja $f : X \subset \mathbb{R} \rightarrow X$, onde X é um compacto, f contínua. Então, a entropia topológica de f em X é definida como:*

$$h(f, X) = \lim_{\epsilon \rightarrow 0} s(\epsilon, X).$$

Como a entropia topológica mede a complexidade das órbitas de um mapa, seria razoável esperar que ela fosse preservada por conjugações topológicas. De fato, é possível demonstrar o teorema:

Teorema 1.4.4 *Se $f : X \subset \mathbb{R} \rightarrow X$ e $g : Y \subset \mathbb{R} \rightarrow Y$ são funções contínuas e se $h : X \rightarrow Y$ é uma conjugação topológica entre f e g , então a entropia topológica de f em X é igual à de g em Y .*

Este teorema será usado no capítulo 2 para obter a entropia topológica do mapa quadrático via dinâmica simbólica, o que simplifica consideravelmente a demonstração.

CAPÍTULO 2

Dinâmica do mapa quadrático

Pode-se fazer uma tentativa de modelar o crescimento de uma população por meio de uma função unidimensional: dado o número inicial de indivíduos, itera-se a função sobre esse valor, e cada iteração corresponde a uma unidade de tempo(ver, por exemplo, [4]). O modelo resultante, é claro, é bastante simplificado, mas exibe um grande número de características que se observam em modelos mais complexos.

Para obter uma tal função, observa-se que: (a) o número de indivíduos na iteração seguinte deve ser proporcional ao número atual e (b) deve haver um valor limite para a população, pois caso contrário a população simplesmente divergiria para qualquer valor inicial. Dadas essas condições, uma possível função é $F_\mu(x) = \mu x(1 - x)$, onde $\mu > 0$ é um parâmetro real.

Para $0 < \mu \leq 1$, F_μ tem somente um ponto fixo atrator em $I = [0,1]$,

$x = 0$, e nenhuma órbita periódica; todos os pontos de I tendem, por iteração, a $x = 0$. Entre $\mu = 1$ e $\mu = 4$, a função F_μ passa por uma série de *bifurcações de duplicação de período*, descritas em detalhe em [1], e do comportamento simples da dinâmica para $\mu = 1$ chega-se ao comportamento caótico para $\mu \geq 4$, que é a situação que vamos estudar com detalhes.

O mapa quadrático $F_\mu(x) = \mu x(1 - x)$, para $\mu > 4$, é especialmente "mal comportado": os pontos cujas órbitas não divergem (para $-\infty$) formam um conjunto de medida nula. Por isso, para o estudo dessas órbitas serão necessários métodos indiretos; neste capítulo, descreveremos a dinâmica simbólica, e no capítulo 3, um método analítico para estudar as órbitas e obter algumas delas numericamente.

2.1 Conjunto invariante

Os pontos fixos de F_μ são $p_0 = 0$ e $p_1 = 1 - \frac{1}{\mu}$, ambos repulsores, pois

$$\begin{aligned} |F'_\mu(0)| &= \mu > 4 \\ |F'_\mu(1 - \frac{1}{\mu})| &= |2 - \mu| > 2 \end{aligned}$$

As órbitas periódicas estão contidas no conjunto invariante, assim é natural determinar primeiro esse conjunto.

Lema 2.1.1 *Se $x \notin [0, 1]$, então $\lim_{n \rightarrow \infty} F_\mu^n(x) = -\infty$.*

Prova. Se $x < 0$, $F_\mu(x) < x$; logo, a sequência $\{x, F_\mu(x), F_\mu^2(x), \dots\}$ é estritamente decrescente, e não pode ser limitada inferiormente, pois nesse caso, se $k = \inf\{F_\mu^n(x)\}_{n \in \mathbb{N}}$, teríamos

$$F_\mu(k) < k \Rightarrow \inf\{F_\mu^n(x)\}_{n \in \mathbb{N}} < k,$$

o que é absurdo.

Se $x > 1$, $F_\mu(x) < 0$, e recaímos no caso anterior. $\square$

Pelo lema 2.1.1, o conjunto invariante está contido no intervalo $I = [0, 1]$.

O conjunto invariante é o conjunto Λ_μ dos pontos x tais que $F_\mu^n(x) \in I$, para todo $n = 0, 1, 2, 3, \dots$. Assim, podemos obter Λ_μ removendo de I os pontos tais que $F_\mu^n(x) \notin I$ para algum n .

O conjunto $A_0 = \{x \in I \mid F_\mu(x) \notin I\}$ é um intervalo aberto contido em I , como se vê na figura abaixo, e $I - A_0$ é constituído de dois intervalos, I_0 e I_1 .

Seja agora $A_1 = \{x \in I \mid F_\mu(x) \in A_0\} = \{x \in I \mid F_\mu^2(x) \notin I\}$, e assim por diante, $A_{n+1} = \{x \in I \mid F_\mu(x) \in A_n\}$, $n = 0, 1, 2, \dots$, recursivamente. Então, o conjunto

$$\Lambda_\mu = I - \bigcup_{n=0}^{\infty} A_n$$

é o conjunto invariante de F_μ .

Vejamos, geometricamente, como é o conjunto Λ_μ , analisando os passos da construção recursiva acima.

A_0 é um intervalo aberto "centrado" em I , tal que $I - A_0$ é a união dos dois intervalos fechados I_0 e I_1 , como abaixo:

A_1 é a pré-imagem de A_0 . Como F_μ é monótona e contínua em I_0 e I_1 , e $F_\mu(I_0) = F_\mu(I_1) = I$, vemos que a pré-imagem de A_0 é constituída de dois intervalos abertos simétricos (em relação a $x = \frac{1}{2}$), um em I_0 e outro em I_1 . Logo, $I - A_0 \cup A_1$ é a união de quatro intervalos fechados I_{00} , I_{01} , I_{10} e I_{11} , os dois primeiros contidos em I_0 e os dois últimos em I_1 , como na figura:

Continuando esse argumento, vemos que a construção do conjunto Λ_μ é semelhante à construção do conjunto de Cantor (de razão $1/3$). De fato, com a definição abaixo, Λ_μ é um conjunto *do tipo Cantor*.

Definição 2.1.2 Um conjunto $E \subset \mathbb{R}$ é um conjunto de Cantor se E é fechado, não contém intervalos e todo ponto de E é ponto de acumulação de pontos do conjunto.

(A demonstração de que Λ_μ é um conjunto de Cantor pode ser encontrada em [1].)

Os passos seguintes, ou seja, a análise dos pontos periódicos e a demonstração de que F_μ é caótica (segundo a def. 1.2.3), exigirão a introdução da dinâmica simbólica.

2.2 Dinâmica simbólica

Definição 2.2.1 Seja Σ_2 o espaço das seqüências infinitas de 0's e 1's, ou seja,

$$\Sigma_2 = \{(s_0 s_1 s_2 \dots) \mid s_i \in \{0, 1\}, i \in \mathbb{N}\}.$$

Definimos em Σ_2 a métrica dada por

$$d(s, t) = \sum_{i=0}^{\infty} \frac{|s_i - t_i|}{2^i},$$

onde $s, t \in \Sigma_2$.

Proposição 2.2.2 d é uma métrica em Σ_2 .

Prova . É direto que $d(s, t) \geq 0$ e $d(s, t) = 0 \iff s = t$. Como $|s_i - t_i| = |t_i - s_i|$, segue também que $d(s, t) = d(t, s)$. Para verificar a desigualdade triangular, basta observar que $|r_i - t_i| \leq |r_i - s_i| + |s_i - t_i|$, o que implica que $d(r, t) \leq d(r, s) + d(s, t)$. $\square$

Nesta métrica, duas seqüências são próximas se elas coincidem nas primeiras posições; mais precisamente,

Proposição 2.2.3 Sejam $s, t \in \Sigma_2$. Se $s_i = t_i$, $i = 0, 1, \dots, n$, então $d(s, t) \leq \frac{1}{2^n}$. Reciprocamente, se $d(s, t) < \frac{1}{2^n}$, então $s_i = t_i$, $i = 0, 1, \dots, n$.

Prova . Seja $s_i = t_i$, $i = 0, \dots, n$. Nesse caso,

$$d(s, t) = \sum_{i=n+1}^{\infty} \frac{|s_i - t_i|}{2^i}$$

Como $|s_i - t_i| \leq 1$, a soma acima está limitada pela série geométrica

$$\sum_{i=n+1}^{\infty} \frac{1}{2^i} = \frac{1}{2^n}$$

Logo, $d(s, t) \leq \frac{1}{2^n}$.

Seja, por outro lado, $d(s, t) < \frac{1}{2^n}$. Se ocorrer $s_k \neq t_k$ para algum $k \leq n$, então

$$d(s, t) = \sum_{i=1}^{\infty} \frac{1}{2^i} \geq \frac{1}{2^k} \geq \frac{1}{2^n},$$

absurdo. Logo $s_i = t_i$, $i = 0, \dots, n$. $\square$

Vamos agora definir um mapa sobre Σ_2 , o *shift* (para a esquerda), cuja ação é simplesmente eliminar o primeiro dígito da seqüência à qual é aplicado, ou, visto de outro modo, deslocar a seqüência um dígito para a esquerda (desprezando o primeiro dígito), como o *shift* usual sobre números binários.

Definição 2.2.4 *Seja $s = (s_0 s_1 s_2 \dots) \in \Sigma_2$; definimos a função $\sigma: \Sigma_2 \rightarrow \Sigma_2$ tal que*

$$\sigma(s) = (s_1 s_2 s_3 \dots),$$

o shift sobre Σ_2 .

Observe-se que o *shift* não é um mapa inversível, pois a informação sobre s_0 é perdida ao se aplicar σ . Mais adiante, veremos que o *shift* é topologicamente conjugado ao mapa quadrático restrito ao seu conjunto invariante; isso tornará o estudo da dinâmica de F_μ muito mais simples.

Os pontos fixos de σ são as seqüências que só têm 0's ou só têm 1's, ou seja,

$$(0000\dots) \text{ e }$$

$$(1111\dots).$$

Os pontos periódicos de σ são as seqüências em que um certo bloco finito de 0's e 1's se repete ao infinito; os pontos periódicos de período n podem então ser determinados encontrando todos os possíveis blocos de tamanho n ; resumindo,

$$(s_0 s_1 \dots s_n s_0 s_1 \dots s_n s_0 s_1 \dots s_n \dots), \text{ onde } s_k = 0 \text{ ou } 1, \text{ para } k = 0, \dots, n.$$

Conclui-se que σ tem 2^n pontos periódicos de período n , para qualquer $n \geq 0$.

Vamos agora mostrar que σ é caótica, ou seja, que verifica os 3 itens da def.

1.2.3:

(1) σ é topologicamente transitivo.

Prova . Para provar (1), basta encontrar uma órbita densa em Σ_2 . Lembremos que, em Σ_2 , duas seqüências são tão mais próximas quanto maior for o número de dígitos que coincidem no início das mesmas.

Seja a seqüência construída do seguinte modo: primeiro, listam-se todos os blocos de 0's e 1's de tamanho 1; depois, os blocos de tamanho 2; e assim por diante, como abaixo:

$$r = (0 \ 1 \mid 00 \ 01 \ 10 \ 11 \mid 000 \ 001 \ 010 \ 011 \ \dots \mid \dots)$$

Seja agora uma seqüência qualquer $s = (s_0 s_1 s_2 \dots) \in \Sigma_2$, e $\delta > 0$. Se $n > 0$ é um inteiro tal que $\frac{1}{2^n} < \delta$, o bloco $(s_0 s_1 \dots s_n)$, de tamanho $n + 1$, certamente está listado na seqüência r . Logo, basta aplicar σ um número suficiente de vezes sobre r para que esse bloco fique no início; ou seja, existe $k > 0$ tal que

$$\sigma^k(r) = (s_0 s_1 s_2 \dots s_n \dots).$$

e, usando a proposição 2.2.3, obtemos

$$d(s, \sigma^k(r)) \leq \frac{1}{2^n} < \delta.$$

Resumindo: dado $s \in \Sigma_2$ qualquer, e $\delta > 0$, existe $k > 0$ tal que $d(s, \sigma^k(r)) < \delta$, isto é, a órbita $\{r, \sigma(r), \sigma^2(r), \dots\}$ é densa em Σ_2 .

Observe-se que, se mudarmos a ordem na qual listamos os blocos em r , obtemos outra seqüência em Σ_2 cuja órbita é densa. Como há infinitas possibilidades, temos na verdade infinitas seqüências cuja órbita é densa em Σ_2 . $\square$

(2) Os pontos periódicos de σ são densos em Σ_2 .

Prova . Seja uma seqüência qualquer $s = (s_0 s_1 s_2 \dots) \in \Sigma_2$, e $\delta > 0$. Se $n > 0$ é um número inteiro tal que $\frac{1}{2^n} < \delta$, a seqüência $r = (s_0 s_1 \dots s_n s_0 s_1 \dots s_n \dots)$, formada pela repetição ao infinito do bloco $(s_0 s_1 \dots s_n)$, é periódica para σ , e, novamente usando a

proposição 2.2.3,

$$d(s, r) \leq \frac{1}{2^n} < \delta.$$

Logo, os pontos periódicos de σ são densos em Σ_2 . $\square$

(3) σ tem dependência sensitiva em relação às condições iniciais.

Prova . Consideraremos aqui δ da definição 1.2.2 como sendo igual a 1. Dado $s = (s_0 s_1 s_2 \dots) \in \Sigma_2$, uma ϵ -vizinhança V_ϵ de s é dada por:

$$V_\epsilon = \{r \in \Sigma_2 \mid r_i = s_i, \text{ para } i = 0, \dots, n, \text{ onde } \frac{1}{2^n} < \epsilon\}$$

(ver a proposição 2.2.3).

Existe um sequência t em V_ϵ tal que $t_{n+1} \neq s_{n+1}$; então,

$$d(\sigma^{n+1}(s), \sigma^{n+1}(t)) = d((s_{n+1}, s_{n+2}, \dots), (t_{n+1}, t_{n+2}, \dots)) > \frac{1}{2^0} = 1$$

como queríamos. $\square$

Portanto, temos um mapa caótico sobre Σ_2 . Agora, vamos obter uma conjugação topológica entre σ e F_μ , e isso já será suficiente para provar que F_μ é caótica; além disso, a dinâmica de F_μ terá as propriedades já encontradas para o *shift*, pelo menos quanto a pontos fixos e periódicos, como vimos na proposição 1.1.14.

Seja um ponto x em Λ_μ . Como vimos no processo de construção de Λ_μ , este pode ser dividido em duas componentes: uma contida em I_0 e outra contida em I_1 . A idéia da aplicação *itinerário* $S : \Lambda_\mu \rightarrow \Sigma_2$ é que, dada a órbita $O_{F_\mu}^+(x) = \{x, F_\mu(x), F_\mu^2(x), \dots\}$, associa-se ao ponto $F_\mu^n(x)$ o valor 0 se ele está em I_0 , e 1 se está em I_1 ;

Definição 2.2.5 A aplicação itinerário de Λ_μ em Σ_2 é definida por:

$$\begin{aligned} S(x) &= (s_0 s_1 s_2 \dots), \text{ onde} \\ s_j &= 0 \text{ se } F_\mu^j(x) \in I_0, \text{ e} \\ s_j &= 1 \text{ se } F_\mu^j(x) \in I_1. \end{aligned}$$

Proposição 2.2.6 A função S é uma conjugação topológica entre F_μ e σ .

Prova . Pela definição,

$$\sigma \circ S(x) = \sigma(s_0 s_1 s_2 \dots) = (s_1 s_2 s_3 \dots)$$

e

$$S \circ F_\mu(x) = S(F_\mu(x)) = (s_1 s_2 s_3 \dots),$$

onde a última igualdade vem do fato de que a órbita de $F_\mu(x)$ é a mesma de x , exceto pela ausência do ponto x no início.

Basta provar que S é um homeomorfismo. Essa prova pode ser encontrada em [1]. $\square$

Assim, concluímos que F_μ é caótica, e, o que nos interessará no capítulo 3, que F_μ tem 2^n pontos periódicos de período n , para todo $n > 0$, e tem infinitas órbitas densas em Λ_μ . O que veremos no capítulo 3 é uma forma de obter resultados numéricos da dinâmica simbólica aqui descrita.

2.3 Entropia

Podemos agora calcular a entropia topológica do *shift* em Σ_2 , com a métrica definida em 2.2.1.

Dado $\epsilon > 0$, os pontos ϵ -separados, isto é, os pontos $x, y \in \Sigma_2$ tais que $d(x, y) > \epsilon$, são aqueles que diferem em alguma das primeiras $m(\epsilon)$ posições, onde $m(\epsilon)$ é o menor inteiro tal que $\frac{1}{2^{m(\epsilon)}} > \epsilon$.

Dois pontos $x, y \in \Sigma_2$ são (n, ϵ) -separados se $\sigma^j(x)$ e $\sigma^j(y)$ são ϵ -separados para algum $0 \leq j < n$. Logo, x e y são (n, ϵ) -separados se $x_k \neq y_k$ para algum $0 \leq k < n + m(\epsilon)$. Segue que o conjunto maximal (n, ϵ) -separado tem $2^{n+m(\epsilon)}$ pontos. Assim,

$$s(\epsilon, \Sigma_2) = \limsup_{n \rightarrow \infty} \frac{1}{n} \log 2^{n+m(\epsilon)} = \limsup_{n \rightarrow \infty} \frac{1}{n} (n \log 2 + m(\epsilon) \log 2) = \log 2.$$

Finalmente,

$$h(\sigma, \Sigma_2) = \lim_{\epsilon \rightarrow 0} s(\epsilon, \Sigma_2) = \log 2.$$

Pelo teorema 1.4.4, a entropia do mapa quadrático no conjunto invariante é $h(F_\mu, \Lambda_2) = \log 2$.

CAPÍTULO 3

Mapa quadrático: análise numérica

Neste capítulo propomos um método para a obtenção dos pontos periódicos e das órbitas densas do mapa quadrático para $\mu > 4$. Esse método pode ser visto como uma versão analítica da dinâmica simbólica, e esse paralelo é descrito com mais detalhes no final do capítulo, onde também exibimos alguns resultados numéricos.

3.1 Pontos periódicos

Apesar de o mapa quadrático $F_\mu(x)$ não ser inversível, podemos definir duas inversas locais, dadas pelos dois ramos da solução de $F_\mu(y) = x$ em função de x .

Definição 3.1.1 *Definimos as inversas locais de F_μ em $[0, 1]$:*

$$\begin{aligned} F_0 : [0, 1] &\rightarrow [0, 1] \text{ tal que } x \mapsto \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{x}{\mu}} \\ F_1 : [0, 1] &\rightarrow [0, 1] \text{ tal que } x \mapsto \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{x}{\mu}} \end{aligned}$$

Na definição das inversas, o parâmetro fica subentendido, já que sempre estaremos considerando um único μ fixo. Observe-se que a imagem de F_0 é I_0 e a imagem de F_1 é I_1 (ver a definição de I_0 e I_1 no cap.2).

Lema 3.1.2 Se $x \in I_0$, então $F_0(F_\mu(x)) = x$; se $x \in I_1$, então $F_1(F_\mu(x)) = x$.

Prova. De acordo com a definição 3.1.1, F_0 é a inversa de F_μ em I_0 ; logo, o resultado é direto. O mesmo para F_1 . $\square$

Definição 3.1.3 Dada uma n -upla ordenada $(i_1, \dots, i_n)$, onde $i_k = 0$ ou 1 , $k = 1, \dots, n$, definimos as aplicações

$H_{i_1 \dots i_n} : [0, 1] \rightarrow [0, 1]$ como sendo

$$H_{i_1 \dots i_n} = F_{i_1} \circ \dots \circ F_{i_n}.$$

ou seja, composições de inversas de F_μ . (veja abaixo o caso $n = 2$)

Observe-se que, como as inversas F_k são funções de $[0, 1]$ em $[0, 1]$, as composições acima estão bem definidas para todo $x \in [0, 1]$.

Existem 2^n funções $H_{i_1 \dots i_n}$ para cada $n > 0$, e vamos ver que os pontos fixos dessas 2^n funções são exatamente os pontos periódicos de período n de F_μ .

Lema 3.1.4 *Para todo $x \in [0, 1]$, e toda n -upla ordenada $(i_1, \dots, i_n)$, onde $i_k = 0$ ou 1 , $k = 1, \dots, n$, verifica-se*

$$F_\mu^n(H_{i_1 \dots i_n}(x)) = x.$$

Prova. A prova será feita por indução sobre n .

Para $n = 1$, as compostas $H_{i_1 \dots i_n}$ se reduzem às inversas F_0 e F_1 , assim basta ver a definição das inversas.

Suponhamos que o lema é verdadeiro para $n = k - 1$, $k \geq 2$. Vamos provar que também é para $n = k$. Temos:

$$F_\mu^k(H_{i_1 \dots i_k}(x)) = F_\mu^{k-1}(F_\mu(H_{i_1 \dots i_k}(x))).$$

Mas $H_{i_1 \dots i_k}(x) = F_{i_1}(H_{i_2 \dots i_k}(x))$ (ver def. 3.1.3). Logo,

$$F_\mu^k(H_{i_1 \dots i_k}(x)) = F_\mu^{k-1}(F_\mu(F_{i_1}(H_{i_2 \dots i_k}(x)))).$$

Novamente pela definição de F_0 e F_1 ,

$$F_\mu(F_{i_1}(H_{i_2 \dots i_k}(x))) = H_{i_1 \dots i_k}(x) \Rightarrow F_\mu^k(H_{i_1 \dots i_k}(x)) = F_\mu^{k-1}(H_{i_1 \dots i_k}(x)).$$

Usando a hipótese de indução, vem finalmente:

$$F_\mu^k(H_{i_1 \dots i_k}(x)) = x. \quad \square$$

Corolário 3.1.5 *Se $x \in [0, 1]$ é ponto fixo de $H_{i_1 \dots i_n}$ ($n \geq 1$), então x é ponto periódico de período n de F_μ .*

Podemos usar as inversas locais para a construção do conjunto invariante Λ_μ como no capítulo 2. Basta observar que os intervalos $H_{i_1 \dots i_n}([0, 1])$, para um dado n ,

são exatamente os intervalos cujos pontos permanecem em $[0, 1]$ após n iterações da função F_μ , já que, pelo lema acima, $F_\mu^n(H_{i_1 \dots i_n}([0, 1])) = [0, 1]$. Definindo

$$I_{i_1 \dots i_n} = H_{i_1 \dots i_n}([0, 1]),$$

obtemos o resultado:

$$\Lambda_\mu = \bigcap_{n=1}^{\infty} \bigcup_{(i_1, \dots, i_n)} I_{i_1 \dots i_n} = \bigcap_{n=1}^{\infty} \bigcup_{(i_1, \dots, i_n)} H_{i_1 \dots i_n}([0, 1]), \text{ com } i_k \in \{0, 1\}, \ k = 1, \dots, n. \quad (3.1)$$

Esse processo de construção de Λ_μ pode ser representado graficamente, como nas três figuras abaixo:

Passo $n = 1$:

Passo $n = 2$:

Passo $n = 3$:

Essa construção será importante na seção seguinte deste capítulo, para analisar órbitas densas de F_μ .

Lema 3.1.6 *Cada função $H_{i_1 \dots i_n}$, $n \geq 1$, tem um único ponto fixo $p_{i_1 \dots i_n} \in [0, 1]$.*

Prova . $H_{i_1 \dots i_n}$ é uma função de $[0, 1]$ em $[0, 1]$, ou seja, $H_{i_1 \dots i_n}([0, 1]) \subset [0, 1]$, e é uma função contínua, pois é composição de funções contínuas (F_0 e F_1). Logo, $H_{i_1 \dots i_n}$ tem pelo menos um ponto fixo $p \in [0, 1]$. Além disso, pelo Corolário 3.1.5, p é ponto periódico de F_μ , de período n .

Os pontos periódicos de F_μ de período n são as soluções da equação $F_\mu^n(x) = x$, que é uma equação polinomial de grau 2^n . Portanto há *no máximo* 2^n pontos periódicos de período n de F_μ . Reunindo esse fato com o obtido acima, vemos que, se provarmos que os pontos fixos das 2^n funções $H_{i_1 \dots i_n}$ são todos distintos, obtemos a unicidade. Para isso, vamos provar que as imagens das funções $H_{i_1 \dots i_n}$ são disjuntas, isto é:

$$H_{i_1 \dots i_n}([0, 1]) \cap H_{j_1 \dots j_n}([0, 1]) = \emptyset,$$

se $(i_1, \dots, i_n) \neq (j_1, \dots, j_n)$ como n -uplas ordenadas.

Usando indução sobre n :

Para $n = 1$, temos $H_0 = F_0$ e $H_1 = F_1$. Nesse caso, pela própria definição de F_0 e F_1 suas imagens são disjuntas, para $\mu > 4$.

Suponhamos, agora, que a afirmação é válida para $n = k - 1$, $k \geq 2$. Vamos provar que é também para $n = k$.

Sejam os intervalos $I_{i_1 \dots i_k}$ definidos como:

$$I_{i_1 \dots i_k} = H_{i_1 \dots i_k}([0, 1]).$$

Observe-se que podemos dividir os mesmos em dois grupos, os contidos em I_0 e os contidos em I_1 :

$$I_{0 \dots i_k} = H_{0 \dots i_k}([0, 1]) \subset I_0$$

$$I_{1 \dots i_k} = H_{1 \dots i_k}([0, 1]) \subset I_1$$

Basta então provar que os intervalos de cada grupo são disjuntos. Mas

$$H_{0 \dots i_k}([0, 1]) = F_0(H_{i_2 \dots i_k}([0, 1]))$$

$$H_{1 \dots i_k}([0, 1]) = F_1(H_{1 \dots i_k}([0, 1]))$$

Usando a hipótese de indução, os conjuntos $H_{i_2 \dots i_k}([0, 1])$ são disjuntos. Como F_0 e F_1 são contínuas e monótonas, as imagens de intervalos disjuntos por essas funções são disjuntas, e isso conclui a prova. $\square$

Teorema 3.1.7 *Os pontos periódicos de período n de F_μ são os pontos fixos das 2^n funções $H_{i_1 \dots i_n}$, e podem ser obtidos por:*

$$p_{i_1 \dots i_n} = \lim_{k \rightarrow \infty} H_{i_1 \dots i_n}^k(x), \text{ para qualquer } x \in [0, 1].$$

(ou seja, são pontos fixos atratores das $H_{i_1 \dots i_n}$).

Prova. A primeira afirmação já foi demonstrada; basta mostrar que os pontos fixos das funções $H_{i_1 \dots i_n}$ são atratores.

Consideremos a sequência de intervalos

$$\{H_{i_1 \dots i_n}^k([0, 1])\}_{k \in \mathbb{N}}.$$

Como $H_{i_1 \dots i_n}([0, 1]) \subset [0, 1]$ e $H_{i_1 \dots i_n}$ é contínua e monótona, a sequência dada é uma sequência de intervalos fechados encaixantes, isto é $H_{i_1 \dots i_n}^{k+1}([0, 1]) \subset H_{i_1 \dots i_n}^k([0, 1])$. Logo, pode ocorrer um dos dois casos:

- (a) $\lim_{k \rightarrow \infty} H_{i_1 \dots i_n}^k([0, 1]) = p_{i_1 \dots i_n}$ (ponto fixo atrator);
- (b) $\lim_{k \rightarrow \infty} H_{i_1 \dots i_n}^k([0, 1]) = [a, b] \subset [0, 1]$, $a < b$.

Suponhamos que ocorresse (b). Nesse caso, teríamos:

$$H_{i_1 \dots i_n}([a, b]) = [a, b].$$

Como $H_{i_1 \dots i_n}$ tem um único ponto fixo, não podemos ter $H_{i_1 \dots i_n}(a) = a$ e $H_{i_1 \dots i_n}(b) = b$. Logo,

$$H_{i_1 \dots i_n}(a) = b$$

e

$$H_{i_1 \dots i_n}(b) = a.$$

Mas daí vem:

$$H_{i_1 \dots i_n}(H_{i_1 \dots i_n}(a)) = H_{i_1 \dots i_n i_1 \dots i_n}(a) = a$$

e

$$H_{i_1 \dots i_n}(H_{i_1 \dots i_n}(b)) = H_{i_1 \dots i_n i_1 \dots i_n}(b) = b,$$

ou seja: a função $H_{i_1 \dots i_n i_1 \dots i_n} = H_{i_1 \dots i_n} \circ H_{i_1 \dots i_n}$ tem dois pontos fixos, o que contradiz o lema 3.1.6. Portanto, devemos ter (a), o que conclui a prova. $\square$

3.2 Órbitas densas em Λ_μ

Como vimos no capítulo 2, F_μ tem órbitas densas em Λ_μ . Podemos usar as inversas para obter tais órbitas; a idéia é obter órbitas que tenham intersecção não-nula com *todos* os intervalos $I_{i_1 \dots i_n}$ definidos na seção anterior, para cada $n > 1$. Ou seja, no passo n da construção de Λ_μ descrita na seção anterior, a órbita deve passar por todos os intervalos $I_{i_1 \dots i_n}$. Se provarmos que os diâmetros $d(I_{i_1 \dots i_n})$ tendem a zero quando $n \rightarrow \infty$, estará provado que a órbita é densa, pois nesse caso os pontos dos intervalos $I_{i_1 \dots i_n}$ "tendem" à órbita quando $n \rightarrow \infty$.

Proposição 3.2.1 *Os diâmetros $d(I_{i_1 \dots i_n})$ tendem a zero quando $n \rightarrow \infty$.*

Prova. Suponhamos que isso não ocorresse. Então, existiria um intervalo contido em Λ_μ . Mas isso não pode ocorrer, pois Λ_μ é um conjunto de Cantor. $\square$

A proposição acima depende, é claro, da prova de que Λ_μ é um conjunto de Cantor. Uma demonstração pode ser encontrada em [1].

Podemos agora encontrar a órbita densa. Seja a órbita obtida aplicando a $x = 1$ todas as funções $H_{i_1}(x)$ (ou seja, $H_0 = F_0$ e $H_1 = F_1$); em seguida, todas as funções $H_{i_1 i_2}(x)$ (ou seja, $H_{00} = F_0 \circ F_0$, $H_{01} = F_0 \circ F_1$, $H_{10} = F_1 \circ F_0$ e $H_{11} = F_1 \circ F_1$); e assim por diante, obtendo a órbita:

$$O = \{1, H_0(1), H_1 \circ H_0(1), H_{00} \circ H_1 \circ H_0(1), H_{01} \circ H_{00} \circ H_1 \circ H_0(1), \dots\},$$

Em primeiro lugar, observe-se que os pontos da órbita realmente estão em Λ_μ , pois são imagens inversas n -ésimas de $x = 1$, e $F_\mu(1) = 0$.

Lembrando que os intervalos $I_{i_1 \dots i_n}$ são as imagens das funções $H_{i_1 \dots i_n}$, vemos que a órbita passa por todos os $I_{i_1 \dots i_n}$, já que todas as funções $H_{i_1 \dots i_n}$ são aplicadas em algum momento.

Seja agora um ponto qualquer $x \in \Lambda_\mu$. Pela proposição 3.2.1, dado qualquer $\epsilon > 0$ podemos encontrar um inteiro $n > 0$ tal que $d(I_{i_1 \dots i_n}) < \epsilon$, para todas as n -uplas $(i_1, \dots, i_n)$. Como $x \in \Lambda_\mu$, x está em algum dos intervalos $I_{i_1 \dots i_n}$. Como vimos, existe também um ponto p da órbita O em $I_{i_1 \dots i_n}$; logo, $d(p, x) < \epsilon$ e concluímos o argumento, e podemos enunciar a

Proposição 3.2.2 *A órbita*

$$O = \{1, H_0(1), H_1 \circ H_0(1), H_{00} \circ H_1 \circ H_0(1), H_{01} \circ H_{00} \circ H_1 \circ H_0(1), \dots\},$$

descrita acima, é densa em Λ_μ .

Como no caso da dinâmica simbólica, podemos aplicar as funções $H_{i_1 \dots i_n}$ em outra ordem, e fazendo isso obtemos infinitas órbitas densas.

3.3 Caos

Usando os métodos desenvolvidos nas seções anteriores, podemos provar que o mapa quadrático tem comportamento caótico para $\mu > 4$ sem recorrer à dinâmica simbólica.

(1) F_μ tem órbitas densas em Λ_μ .

Prova . Ver a seção anterior.

(2) Os pontos periódicos de F_μ são densos em Λ_μ .

Prova . Os pontos periódicos de período n , como vimos na seção 3.1, podem ser obtidos por:

$$p_{i_1 \dots i_n} = \lim_{k \rightarrow \infty} H_{i_1 \dots i_n}^k(x), \text{ para qualquer } x \in [0, 1].$$

Seja um número real $\epsilon > 0$. Como vimos, os diâmetros dos intervalos

$$I_{i_1 \dots i_n} = H_{i_1 \dots i_n}([0, 1])$$

tendem a zero quando $n \rightarrow \infty$. Logo, podemos encontrar $n > 0$ tal que

$$d(I_{i_1 \dots i_n}) < \epsilon, \\ \text{para toda } n\text{-upla } (i_1, \dots, i_n).$$

Qualquer ponto $x \in \Lambda_\mu$ deve estar em algum dos intervalos $I_{i_1 \dots i_n}$. Por outro lado, existe um ponto periódico de período n em cada um dos intervalos $I_{i_1 \dots i_n}$, que é o ponto fixo da função $H_{i_1 \dots i_n}$. Logo, dado qualquer ponto x de Λ_μ , existe um ponto periódico (de F_μ) $p_{i_1 \dots i_n}$ tal que

$$d(p_{i_1 \dots i_n}, x) < d(I_{i_1 \dots i_n}) < \epsilon$$

(onde foi usada a mesma notação para *distância* entre dois pontos, $d(x, y)$, e *diâmetro* de um intervalo, $d(I)$). $\square$

(3) F_μ tem dependência sensitiva em relação às condições iniciais.

Prova . Seja o valor de δ da definição 1.2.2 menor que a distância entre os intervalos I_0 e I_1 , $\delta < d(I_0, I_1)$. Sejam, também, um ponto qualquer $x \in \Lambda_\mu$, e $V_\epsilon = \{y \in R \mid d(y, x) < \epsilon\}$ uma vizinhança de x , para algum número real $\epsilon > 0$. Como já vimos, para todo $n > 0$, todo ponto de Λ_μ está contido em algum dos intervalos $I_{i_1 \dots i_n}$. Seja então n tal que

$$d(I_{i_1 \dots i_n}) < \epsilon,$$

para toda n -upla $(i_1, \dots, i_n)$.

Então, o ponto x está contido em algum dos intervalos $I_{i_1 \dots i_n}$. Como os intervalos $I_{i_1 \dots i_n}$ formam uma sequência encaixante de intervalos, os intervalos $I_{i_1 \dots i_n 0}$ e $I_{i_1 \dots i_n 1}$ (ou seja, os intervalos associados às n -uplas $(i_1, \dots, i_n, 0)$ e $(i_1, \dots, i_n, 1)$) estão contidos em $I_{i_1 \dots i_n}$. Novamente, o ponto x deve estar em alguns dos intervalos $I_{i_1 \dots i_{n+1}}$. Como já está em $I_{i_1 \dots i_n}$, temos

$$x \in I_{i_1 \dots i_n 0}$$

ou

$$x \in I_{i_1 \dots i_n 1}.$$

Suponhamos que ocorra o primeiro caso. Então, seja $y \in I_{i_1 \dots i_n 1}$. Temos:

$$F_\mu^n(x) \in I_0$$

e

$$F_\mu^n(y) \in I_1,$$

logo

$$d(F_\mu^n(x), F_\mu^n(y)) \geq d(I_0, I_1) > \delta. \quad \square$$

Assim, concluímos a prova de que F_μ tem comportamento caótico para $\mu >$

4. $\square$

3.4 Comparação com a dinâmica simbólica

Consideremos a sequência seguinte de conjuntos, onde $I = [0, 1]$:

$$\{I, H_{i_1}(I), H_{i_1 i_2}(I), H_{i_1 i_2 i_3}(I), \dots, H_{i_1 \dots i_n}(I), \dots\}$$

Como cada $H_{i_1 \dots i_n}$ é uma composição de n funções F_0 e F_1 , que são funções monótonas e contínuas, os conjuntos $H_{i_1 \dots i_n}(I)$ são intervalos fechados. Como

$$H_{i_1 \dots i_n}(I) = H_{i_1 \dots i_{n-1}}(F_{i_n}(I))$$

e

$$F_{i_n}(I) \subset I,$$

a sequência de intervalos dada é encaixante. Novamente, como na prova da proposição 3.2.2, o conjunto dado por

$$\bigcap_{n=1}^{\infty} H_{i_1 \dots i_n}(I)$$

está contido no conjunto Λ_μ , que é de Cantor. Logo, deve consistir de um único ponto (e não um intervalo). Concluindo, a cada sequência $(s_0 s_1 s_2 \dots)$ de 0's e 1's associamos um único ponto de Λ_μ . Então, chamando esse ponto de p , vemos que, para qualquer $x \in I$,

$$p = \lim_{n \rightarrow \infty} H_{i_1 \dots i_n}(x).$$

Observemos que:

$$p = H_{i_1}(\lim_{n \rightarrow \infty} H_{i_2 \dots i_n}(x)) = F_{i_1}(\lim_{n \rightarrow \infty} H_{i_2 \dots i_n}(x)).$$

Logo, p está na imagem de F_{i_1} , ou seja, I_{i_1} . Agora, aplicando F_μ , vem:

$$F_\mu(p) = F_\mu(F_{i_1}(\lim_{n \rightarrow \infty} H_{i_2 \dots i_n}(x))) = \lim_{n \rightarrow \infty} H_{i_2 \dots i_n}(x).$$

Novamente,

$$F_\mu(p) = H_{i_2}(\lim_{n \rightarrow \infty} H_{i_3 \dots i_n}(x)) = H_{i_2}(\lim_{n \rightarrow \infty} H_{i_3 \dots i_n}(x)).$$

Logo, $F_\mu(p)$ está em I_{i_2} . Continuando o procedimento, obtemos que $F_\mu^n(p) \in I_{i_n}$. Isto é, o ponto p é exatamente o ponto associado na dinâmica simbólica à sequência $(i_1 i_2 i_3 \dots)$.

Dada então a relação biunívoca entre os pontos de Σ_2 e as funções $H_{i_1 \dots i_n}$:

$$(i_1 i_2 i_3 \dots) \mapsto H_{i_1 i_2 i_3 \dots},$$

a técnica de usar as inversas locais é "equivalente" à dinâmica simbólica.

3.5 Resultados numéricos

Usando o software Mathcad num micro 386, obtivemos alguns pontos periódicos de F_μ , com precisão de 15 dígitos (precisão *double*), e os primeiros estágios da construção de uma órbita densa.

Para períodos muito grandes, começa-se a observar desvios dos resultados por erro de truncamento, assim para tais valores usamos programas escritos especificamente para isso em C++, no compilador da Borland ainda rodando num 386. Foi usada uma biblioteca de classes C++ para aritmética de ponto flutuante de precisão arbitrária; os programas que geram os resultados estão anexos.

Os resultados numéricos obtidos estão listados no apêndice D.

Alguns exemplos de programas para estudo de órbitas em mapas (uni e bidimensionais) podem ser encontrados em [2].

Apêndice A

Observações sobre $HD(\Lambda_\mu)$

Podemos usar os lemas 1.3.5 e 1.3.4 e as inversas locais de F_μ para tentar obter limitações da dimensão de Hausdorff do conjunto invariante do mapa quadrático, para $\mu > 4$. No entanto, para aplicar os lemas precisamos supor que a medida de Hausdorff $H^s(\Lambda_\mu)$, onde s é a dimensão de Hausdorff de Λ_μ , é finita e não-nula.

Não provamos que $0 < H^s(\Lambda_\mu) < \infty$, mas, se supusermos que isso é verdadeiro, podemos enunciar a seguinte conjectura:

Conjectura A.1 *A dimensão de Hausdorff $HD(\Lambda_\mu)$ verifica as seguintes desigualdades:*

- (a) $HD(\Lambda_\mu) \geq \frac{\log 2}{\log \mu}$, para $\mu > 4$
- (b) $HD(\Lambda_\mu) \leq \frac{\log 2}{\log \sqrt{\mu^2 - 4\mu}}$, para $\mu > 2 + \sqrt{5}$.

Prova (possível).

O conjunto invariante Λ_μ satisfaz :

$$F_\mu(\Lambda_\mu) = \Lambda_\mu.$$

Por outro lado, é fácil verificar que

$$F_\mu\left(\frac{1}{2} - x\right) = F_\mu\left(\frac{1}{2} + x\right),$$

ou seja: F_μ é simétrica em relação a $x = \frac{1}{2}$. Isso implica que o conjunto Λ_μ é simétrico em relação a $x = \frac{1}{2}$. Sejam, então, as duas componentes simétricas de Λ_μ :

$$\Lambda_\mu^e = \Lambda_\mu \cap I_0,$$

$$\Lambda_\mu^d = \Lambda_\mu \cap I_1.$$

Novamente pela simetria de F_μ , vem:

$$F_\mu(\Lambda_\mu^e) = F_\mu(\Lambda_\mu^d) = \Lambda_\mu.$$

(ver figura abaixo).

Usando as inversas locais, e lembrando que F_0 é a inversa local em I_0 e F_1 a inversa local em I_1 , vem:

$$F_0(F_\mu(\Lambda_\mu^e)) = F_0(\Lambda_\mu) \Rightarrow \Lambda_\mu^e = F_0(\Lambda_\mu)$$

$$F_1(F_\mu(\Lambda_\mu^d)) = F_1(\Lambda_\mu) \Rightarrow \Lambda_\mu^d = F_1(\Lambda_\mu)$$

(a) A derivada de F_0 em $[0, 1]$ tem o valor mínimo $\frac{1}{\mu}$ em $x = 0$. Logo, pelo teorema do valor médio,

$$|F_0(x) - F_0(y)| \geq \frac{1}{\mu} |x - y|, \forall x, y \in [0, 1].$$

Usando o lema 1.3.5, sendo $s = HD(\Lambda_\mu)$, vem:

$$H^s(F_0(\Lambda_\mu)) \geq \frac{1}{\mu^s} H^s(\Lambda_\mu)$$

Usando as igualdades obtidas acima, $F_0(\Lambda_\mu) = \Lambda_\mu^e$; logo,

$$H^s(\Lambda_\mu^e) \geq \frac{1}{\mu^s} H^s(\Lambda_\mu).$$

Mas, como $\Lambda_\mu = \Lambda_\mu^e \cup \Lambda_\mu^d$ e as componentes são simétricas, $H^s(\Lambda_\mu) = 2H^s(\Lambda_\mu^e) = 2H^s(\Lambda_\mu^d)$. Finalmente,

$$\frac{1}{2} H^s(\Lambda_\mu) \geq \frac{1}{\mu^s} H^s(\Lambda_\mu)$$

Supondo que $H^s(\Lambda_\mu) > 0$, obtemos, dividindo:

$$\frac{\mu^s}{2} \geq 1 \Rightarrow s \geq \frac{\log 2}{\log \mu}. \quad \square$$

(b) A derivada de F_0 em $[0, 1]$ tem o valor máximo $\frac{1}{\sqrt{\mu^2 - 4\mu}}$ em $x = 1$. Novamente pelo teorema do valor médio,

$$|F_0(x) - F_0(y)| \leq \frac{1}{\sqrt{\mu^2 - 4\mu}} |x - y|, \quad \forall x, y \in [0, 1].$$

Usando o lema 1.3.4, vem:

$$H^s(F_0(\Lambda_\mu)) \leq \frac{1}{\sqrt{\mu^2 - 4\mu}^s} H^s(\Lambda_\mu)$$

Novamente, $F_0(\Lambda_\mu) = \Lambda_\mu^e$. Procedendo exatamente como em (a), obtemos:

$$\frac{\sqrt{\mu^2 - 4\mu}^s}{2} \leq 1 \Rightarrow s \leq \frac{\log 2}{\log \sqrt{\mu^2 - 4\mu}}.$$

Aqui, no entanto, devemos observar que $\log \sqrt{\mu^2 - 4\mu} < 0$ se $\sqrt{\mu^2 - 4\mu} < 1$, o que ocorre para $4 < \mu < 2 + \sqrt{5}$. Isso resultaria em uma limitação sem sentido para a dimensão de Hausdorff, pois ela é sempre positiva. Por isso é necessária uma restrição maior sobre μ neste caso, ou seja, $\mu > 2 + \sqrt{5}$. $\square$

Cabe aqui a observação de que é possível provar que, dada uma função racional f , a dimensão de Hausdorff do conjunto de Julia dessa função depende analiticamente (como função de R em R) da função f . Uma prova desse teorema pode ser encontrada em [5]. Esse é o caso para o mapa quadrático, onde Λ_μ é seu conjunto de Julia; ou seja, $HD(\Lambda_\mu)$ depende analiticamente de μ .

Apêndice B

Outras funções

Os resultados obtidos aqui, principalmente o cálculo de pontos periódicos, talvez possam ser estendidos a uma classe um pouco mais ampla, por exemplo às funções unimodais na reta. Em nossas demonstrações no capítulo 3, no entanto, usamos argumentos bastante particulares para a função quadrática, usando o fato de que ela é um polinômio de grau 2.

Independentemente disso, aplicamos o método usado para obter os pontos periódicos do mapa quadrático a outras funções num intervalo da reta real, e apresentamos alguns resultados aqui somente com caráter ilustrativo.

(a) A função $f(x) = k \cdot \text{sen}(x)$, $x \in [0, \pi]$, onde $k > \pi$.

Seqüência	Valor (15 dígitos)
000	7.60922391561526e-037
001	0.222124172026409
010	0.881208412987881
011	0.612011365415778
100	3.08603303075686
101	2.98798647175979
110	2.29805964748721
111	2.47457678736983

Tabela B.1: Pontos periódicos para $k=4$

As inversas locais neste caso são :

$$f_0(x) = \arcsen\left(\frac{x}{k}\right)$$

e

$$f_1(x) = \pi - \arcsen\left(\frac{x}{k}\right)$$

Esta função é unimodal e bastante parecida com a função quadrática, e para ela obtivemos 2^n pontos (aparentemente) periódicos de período n , para cada n escolhido, indicando que uma possível demonstração poderia ser muito semelhante à usada para o mapa quadrático.

Os pontos encontrados, para período 3, estão na tabela B.1.

(b) A função $f(x) = F_\mu(x) \bmod 1$, $x \in [0, 1]$, para $\mu > 4$.

Esta é a função quadrática analisada anteriormente, modificada (pela operação *módulo 1*) para se tornar uma função de $[0, 1]$ em $[0, 1]$. Observe, pelo gráfico da

função, que neste caso precisamos definir 4 inversas locais, duas para os ramos externos e duas para o ramo interno do gráfico. Mais ainda, se μ fosse suficientemente grande (> 8), precisaríamos de um número ainda maior de inversas, pois então teríamos ainda mais ramos (v. figura abaixo).

Suponhamos aqui que $4 < \mu \leq 8$, e assim só temos 4 ramos. Nesse caso, sejam as inversas F, G os ramos externos e as inversas f, g aos ramos internos (v. figura abaixo).

O método de obtenção dos pontos periódicos, como antes, consistiria então de obter pontos fixos de compostas de F, G, f e g . Como há 4 inversas, a suposição natural seria que houvessem 4^n pontos periódicos de período n , para todo $n > 0$.

Para $\mu = 8$, todas as 4 inversas estão definidas para todo $x \in [0, 1]$, portanto podemos como antes compor as funções sem problemas. Apresentamos alguns resul-

Seqüência	Valor (15 dígitos)
ff	0.179805898398896
fg	0.337153045283501
gf	0.787846954716499
gg	0.695194101601104
FF	0.0
FG	0.143237254218789
GF	0.981762745781211
GG	0.875
fF	0.149838320490311
Ff	0.0190943856236329
gG	0.601144047293392
Gg	0.918159053576897
Fg	0.117909720622769
gF	0.832056147243435
fG	0.416961484972862
Gf	0.944836840176702

Tabela B.2: Pontos periódicos para $\mu=8$

tados na tabela B.2, mais exatamente os 16 (possíveis) pontos periódicos de período 2. De fato, fazendo os cálculos esses pontos se comportam como periódicos, na precisão usada (*double*, 15 dígitos).

No caso $\mu < 8$, ainda podemos definir 4 inversas, mas as inversas do meio não estão definidas para todo $x \in [0, 1]$, assim o método não pode ser aplicado de um modo direto.

(c) A função "dente de serra", $f(x) = \begin{cases} 2x, & \text{se } x < \frac{1}{2} \\ 2-2x, & \text{se } x \geq \frac{1}{2} \end{cases}$, $x \in [0, 1]$.

Seqüência	Valor (15 dígitos)
000	8.67361737988404e-019
001	0.222222222222222
010	0.444444444444444
011	0.285714285714286
100	0.888888888888889
101	0.857142857142857
110	0.571428571428571
111	0.666666666666667

Tabela B.3: Pontos periódicos para "dente de serra"

As inversas locais são:

$$f_0(x) = \frac{x}{2}$$

$$e$$

$$f_1(x) = 1 - \frac{x}{2}$$

Na tabela B.3, temos os pontos encontrados para período 3.

Apêndice C

Um exemplo bidimensional

Podemos construir um mapa bidimensional, a partir do mapa quadrático, que é topologicamente conjugado à restrição do mapa de Smale (*horseshoe*) ao seu conjunto invariante Λ (para detalhes sobre a dinâmica do mapa de Smale, ver [1]). Esse mapa, no entanto, não é contínuo, logo o paralelo com o mapa de Smale não pode ser levado muito adiante.

Usando esse modelo do mapa de Smale, podemos usar os métodos numéricos estudados para o mapa quadrático para obter órbitas periódicas e densas em seu conjunto invariante. Mais ainda, como esse modelo é geometricamente quase idêntico ao *horseshoe*, podemos ter um certo grau de visualização do mesmo em computador.

Definição C.1 *Definimos, para $\mu > 4$, o mapa*

$$H : [0, 1] \times [0, 1] \rightarrow [0, 1] \times [0, 1]$$

tal que

$$(x, y) \mapsto \begin{cases} (F_\mu(x), F_0(y)), & \text{para } x \leq \frac{1}{2} \\ (F_\mu(x), F_1(y)), & \text{para } x > \frac{1}{2} \end{cases}.$$

O quadrado $Q = [0, 1] \times [0, 1]$ é levado por H em duas faixas horizontais:

Observe que a metade esquerda de Q , ou seja, $Q_0 = Q \cap [0, \frac{1}{2}] \times [0, 1]$, é levada em $[0, 1] \times [0, p_0]$ e a metade direita, $Q_1 = Q \cap [\frac{1}{2}, 1] \times [0, 1]$, é levada em $[0, 1] \times [p_1, 1]$. Observe ainda que o segmento sobre o eixo x entre $x = \frac{1}{2}$ e $x = 1$ é levado no segmento sobre a reta $y = 1$ entre $x = 0$ e $x = 1$, e que o segmento sobre a reta $y = 1$ entre $x = \frac{1}{2}$ e $x = 1$ é levado no segmento sobre a reta $y = p_1$ entre $x = 0$ e $x = 1$. Esse comportamento de H sobre Q é idêntico ao do *horseshoe* sobre o quadrado Q , exceto que H , sendo descontínuo, divide a imagem em 2 componentes disjuntas.

Fazendo um argumento similar ao usado no caso do *horseshoe*, obtemos que o conjunto invariante de H é:

$$\Lambda_0 = \Lambda_\mu \times \Lambda_\mu \subset Q$$

(onde Λ_μ é o conjunto invariante de F_μ).

O mapa H é inversível, e é fácil verificar que sua inversa é

$$H^{-1} : [0, 1] \times [0, 1] \rightarrow [0, 1] \times [0, 1]$$

tal que

$$(x, y) \mapsto \begin{cases} (F_0(x), F_\mu(y)), & \text{para } y \leq \frac{1}{2} \\ (F_1(x), F_\mu(y)), & \text{para } y > \frac{1}{2} \end{cases}$$

Definindo os conjuntos $V_0 = [0, p_0] \times [0, 1]$, $V_1 = [p_1, 1] \times [0, 1]$, $H_0 = [0, 1] \times [0, p_0]$ e $H_1 = [0, 1] \times [0, p_0]$, podemos repetir o argumento da pág. 181 de [1], e obtemos que a restrição de H a Λ_0 é topologicamente conjugada ao *shift* σ sobre o espaço Σ_2 das seqüências duplamente infinitas de 0's e 1's. Podemos então traçar os diagramas de conjugação abaixo, onde F é o mapa de Smale :

$$\begin{array}{ccc} \Lambda & \xrightarrow{F} & \Lambda \\ S_F \downarrow & & \downarrow S_F \\ \Sigma_2 & \xrightarrow{\sigma} & \Sigma_2 \\ \\ \Sigma_2 & \xrightarrow{\sigma} & \Sigma_2 \\ S_H \downarrow & & \downarrow S_H \\ \Lambda_0 & \xrightarrow{H} & \Lambda_0 \end{array}$$

onde $S_F(p) = (...s_{-2}s_{-1}.s_0s_1s_2...)$, $s_j = k \Leftrightarrow F^j(p) \in V_k$, para $k = 0, 1$; e $S_H(p) = (...s_{-2}s_{-1}.s_0s_1s_2...)$, $s_j = k \Leftrightarrow H^j(p) \in V_k$, para $k = 0, 1$.

Logo, $H|_{\Lambda_0}$ é topologicamente conjugado a $F|_{\Lambda}$.

Apêndice D

Resultados numéricos

Abaixo estão alguns resultados numéricos obtidos para o mapa quadrático para diversos valores de μ (nas tabelas, "seqüência" se refere a uma seqüência $i_1 \dots i_n$ como descrito no capítulo 3, assim 0010 se refere, por exemplo, ao ponto periódico p_{0010} , que é ponto fixo da função H_{0010}).

(Na tabela D.4 foi usada uma precisão de 100 dígitos significativos, e os números foram truncados para apresentação.)

Na série de tabelas D.5 a D.9 estão os 5 primeiros passos da construção da órbita densa descrita na seção 3.2. Começou-se com o valor $x = 1$; cada tabela se refere a um valor de n , para o qual se aplicam as 2^n funções $H_{i_1 \dots i_n}$ sucessivamente, como descrito na seção citada.

Seqüência	Valor (40 dígitos)
000	+2E-40
001	+1.157177567632227266726464714421762863037E-1
010	+4.1033190170697049376619636916152779815299E-1
011	+1.886928738182453869446121763840821679568E-1
100	+9.702581249155315117911243656303615091318E-1
101	+9.504935299138081354830079947744984671928E-1
110	+6.13882371485713017162861500412840055514E-1
111	+7.506234413965087281795511221945137157107E-1

Tabela D.1: Período 3 $\mu=4.01$

Seqüência	Valor (40 dígitos)
00000	+2E-40
00001	+8.7195810817737692886534392391907620487E-3
00010	+3.46606354500038905106840616051374231488E-2
00011	+1.04095748512356660236131440468174363563E-2
00100	+1.341716959588253546932863595687489117926E-1
00101	+8.92296583042429139503413060015938563876E-2
00110	+4.13078745666352738205991136959436453683E-2
00111	+7.86328125013233566284807319671188009479E-2
01000	+4.6584030436905510953714162146240803891526E-1
01001	+4.2061604035855618722042138869570981804601E-1
01010	+3.258835827964353798207746793868018289713E-1
01011	+3.798927423607382416051015177941565192544E-1
01100	+1.588021516023438260436963659540042337786E-1
01101	+2.096586414632138567448601332771517323765E-1
01110	+2.905232701308148783035242929801558199038E-1
01111	+2.352611137639719548670594917877811580441E-1
10000	+9.978207919295488302930956649141444125611E-1
10001	+9.973973221148998103983654630233572290442E-1
10010	+9.772297296760989358774129275108607802964E-1
10011	+9.799904372225621693023506493724614328585E-1
10100	+8.809307277733788477493790585992764091324E-1
10101	+8.940350939594547696876758856280373351311E-1
10110	+9.446527291161277518987110655385068818971E-1
10111	+9.374144408639433805443282184972859420144E-1
11000	+5.3567195327772823732750530824287250006596E-1
11001	+5.7492234180207323809945585234004197336528E-1
11010	+6.644646010472837345425567599394913477687E-1
11011	+6.274002070333240090890300438244032950253E-1
11100	+8.265391935696829079291878804136217263454E-1
11101	+8.058448879790403141072346814643882893838E-1
11110	+7.214524216791746759159217710340167440596E-1
11111	+7.506234413965087281795511221945137157107E-1

Tabela D.2: Período 5 $\mu=4.01$

Seqüência	Valor (60 dígitos)
000000000000000100110	+7.6635103459617186349066605878770667355652888087E-9
000000000000000100111	+7.2409592798071669817195196399270362094156567545E-9
000000000000000101000	+5.0880529285623651230036702988049207060540959598E-9

Tabela D.3: Período 20 $\mu=4.01$

Iteração	Valor
0	+7.240959279807166981719519639927036209415656754574376375191E-9
1	+2.903624650177645951647491315829010538941786215923669155521E-8
2	+1.164353450912781229042424135826446868404875062596571621659E-7
3	+4.669056794516950308806639499672088925272816378403347357639E-7
4	+1.872290900417633921793006955230416327976922752532142766372E-6
5	+7.507872453727116721825682930101398584816115936345133151311E-6
6	+3.010634250316912450243093893574062707808531235139885221022E-5
7	+1.207227988063539275624890083284857939691194815331159095443E-4
8	+4.840399814969311754943782581013853981477424463511800902624E-4
9	+1.940060804040906940338764838936291940449072325381565852158E-3
10	+7.76455084215129967097874660464041622750907110274190790544E-3
11	+3.08940929954074983282977494430268251945205832318422242948E-2
12	+1.20057988533728432819850162941474948395335962967053384441E-1
13	+4.23632712371083895527676429259688310422423223797512385746E-1
14	+9.79113829894611613197640239654389823602987559877349723041E-1
15	+8.20042513949058407354303878809374450927855760569220694726E-2
16	+3.01871012133748366079739106855318806901544033911272245049E-1
17	+8.45087065710049798400885149122761159506690926977942012125E-1
18	+5.24968817489307368237570168908213048380019687752804017202E-1
19	+9.99999998194274490553029453929107203897912996553774084457E-1
20	+7.24095927980716698171951963992703620941565675457437637519E-9

Tabela D.4: Órbita periódica de período 20

Seqüência	Valor
0	+4.21913119055696967237948564943636423603409510577792E-1
1	+8.83528827085953956153295780778337538097770121729499E-1

Tabela D.5: Órbita densa para $\mu=4.1$ passo 1

Seqüência	Valor
00	+8.36407340956389824150716445113150741098331692469748E-2
01	+3.94268812081745444584393645979453269551024110559738E-1
10	+9.72982349480828844418172014971809785052851520018317E-1
11	+8.17137496966159556952652371650567676866936361506737E-1

Tabela D.6: Órbita densa para $\mu=4.1$ passo 2

Seqüência	Valor
000	+1.79449971453643645175293889990780308828072955171826E-2
001	+1.14382144436749129782525942102712163005480331032792E-1
010	+4.11578320275706735862333682573552254272757062577284E-1
011	+2.11435358530661368325549027813987539734294030591755E-1
100	+9.96699975874537789280386513673167428376292415471977E-1
101	+9.56205575797473576991036376675196501815433136734444E-1
110	+6.74919575884379871160623062174924342926274158064934E-1
111	+7.64444879906879525561024889253583919503362725579032E-1

Tabela D.7: Órbita densa para $\mu=4.1$ passo 3

Seqüência	Valor
0000	+3.92361294689318490173688268973127358135929805545693E-3
0001	+2.91420009713984529395603882156416409907096530150271E-2
0010	+1.15618188333796727636941788531819547039954913275588E-1
0011	+4.70149945651697264837339441581361452235642783393430E-2
0100	+4.20838835895975774386090959529878775584946574109747E-1
0101	+3.37169521694688265037132280686424623905483175492232E-1
0110	+1.80970853703574003868061806404372616831549185445429E-1
0111	+2.70823137937477377586430968995019866958714448067571E-1
1000	+9.98943999144809009049357457643696385712781221897582E-1
1001	+9.89284856105331190382818573763062040610447357719577E-1
1010	+9.15423821264270936249296074805943592560882901747226E-1
1011	+9.30781026745164918948298219993800008175194775321844E-1
1100	+6.08671895814729839263186170393201104048144447009542E-1
1101	+6.54201013276857323354146085740034044888369556830602E-1
1110	+8.07883469964553379062261956466240164911683148986560E-1
1111	+7.58854604189277816041744416279541635058544240934166E-1

Tabela D.8: Órbita densa para $\mu=4.1$ passo 4

Seqüência	Valor
00000	+9.46502879388584497887887268176065957285947078169557E-4
00001	+7.18190948071554357210299445189686547352214821444312E-3
00010	+2.92091520330137490982034789029716266058921722540627E-2
00011	+1.09349975236572647963405419180962933787855099708924E-2
00100	+1.16392242482261987045854066570298566542052403572934E-1
00101	+8.59192196264781452605071723819714510092471100465234E-2
00110	+4.40745239094003203546277353249903905932286839797945E-2
00111	+7.33872679192367854668367572720735144180415679191749E-2
01000	+4.21498673875245118137707748719039282993131851406446E-1
01001	+3.94261372850192227956643183584054601785574985873528E-1
01010	+3.21497952079605037151428142905867491104834205542967E-1
01011	+3.62641459928098147494144355491643031576471553188111E-1
01100	+1.73144022248712337534741990067775177383736449178227E-1
01101	+2.07759691003367614679465250954604455314770116826270E-1
01110	+2.78177196976868013488868489019612994692650461836622E-1
01111	+2.33587488129654168193971316784286641422499023456627E-1
10000	+9.99781042399131914731614834082025253093034516020704E-1
10001	+9.97387475377875777899144975026564079073874288853966E-1
10010	+9.79103781618835927156308158786993696536467964459779E-1
10011	+9.81952368587932895522353794334969749581504137714314E-1
10100	+8.94449615949509283397550268624643109249106131483896E-1
10101	+9.02361136247549370567662141208584465729321981688706E-1
10110	+9.47508048520674779672952470051549816886634774058419E-1
10111	+9.39810106655835501634241831278569690064397523046819E-1
11000	+5.86794770294345123289929136546493752890941528612003E-1
11001	+6.02805476369405949501373965871303934603892381460211E-1
11010	+6.74524520076923566060179901062274296702124504060195E-1
11011	+6.44986798346639303207184991173458147706513466710884E-1
11100	+8.23656039906675944221344330709958415056108324897231E-1
11101	+8.05152199860264663742263244589485189033694112335507E-1
11110	+7.33534525211754931212759945032860387014060626107081E-1
11111	+7.566423895567323543161381250566023545510612631469955E-1

Tabela D.9: Órbita densa para $\mu=4.1$ passo 5

Bibliografia

- [1] Devaney, Robert. "*An Introduction to Chaotic Dynamical Systems*", Addison-Wesley, 1987.
- [2] Devaney, Robert. "*Chaos, Fractals, and Dynamics - Computer Experiments In Mathematics* ". Addison-Wesley, 1990.
- [3] Martin, Miguel-Angel. "*FRACTALES: una Introducción a la Teoría de Estructura y Aplicaciones*", Atas do 28o. Seminário Brasileiro de Análise.
- [4] May, Robert M. "*O Frágil Equilíbrio da Vida* ". Ciência Hoje, Vol. 14, no. 80 (1992), 18-25.
- [5] Melo, Wellington. "*Lectures on One-dimensional Dynamics* ", Impa-CNPq, 1985.
- [6] Ruelle, David. "*Repellers for real analytic maps* ", Ergod. Th. & Dynam. Sys. **2** (1982), 99-107.
- [7] Young, Lai-Sang. "*Entropy, Lyapunov Exponents, and Hausdorff Dimension in Differentiable Dynamical Systems*", IEEE Transactions on Circuits and Systems, Vol. 30, no.8 (1983).

PROGRAMAS

Seguem as listagens dos programas em C++ usados para obter os resultados numéricos apresentados no apêndice D.

A biblioteca FLUTPP foi usada para os cálculos, para que os programas pudessem ser escritos em C++, mas, como é observado em FLUTPP.CPP, possui pouco controle de limites; assim, os resultados foram refeitos (e confirmados) usando o software *Mathematica 2.0*. Ela é apresentada aqui como uma possível implementação de aritmética de múltipla precisão em base decimal, usando estritamente os conhecidos algoritmos usados em cálculos manuais (uma espécie de automatização da tarefa de cálculo). Outras implementações podem ser encontradas na literatura, inclusive implementações mais rápidas e com tratamento de erros mais elaborado.

// PERIODPP.CPP - Este programa calcula, para m_i e n recebidos do usuário, os
// 2^n pontos periodicos do mapa quadratico para o dado valor de m_i .

```
#include <stdio.h>
```

```
#include "flutpp.h" // Ponto flutuante (o numero de digitos da mantissa e'  
// definido em FLUTPP.H).
```

```
const smax = 20; // Periodo maximo aceito.
```

```
const itmax = 50; // Numero maximo de iteracoes para obtencao do ponto  
// periodico.
```

```
static Ereal r0_5("0.5"),r0_25("0.25");
```

```
Ereal f(Ereal x, Ereal  $m_i$ , int i) // Inversas locais.
```

```
{  
Ereal y,b;  
if (i==0) // F0(x)  
{  
x=r0_5-raiz(r0_25-x/ $m_i$ );  
}  
else // F1(x)  
{  
x=r0_5+raiz(r0_25-x/ $m_i$ );  
}  
return(x);  
} // f
```

```
int pot(int n) // Retorna  $2^n$ .
```

```
{  
int a,i;  
a=1;  
if (n==0) return(a);  
else  
{  
for (i=1;i<=n;++i) a=2*a;  
return(a);  
}  
} // pot
```

```
void main() // Corpo principal
```

```
{  
int i,j,k,n,p;  
Ereal  $m_i$ ,x,y,x0;  
int index[smax+1]; // Armazena a sequencia de 0's e 1's.
```

```
printf("Entre valor de  $m_i$  (ate %d digitos signif.):",smax-3);
```



```

le_numero(&mi);
printf("\nEntre periodo (1-%d):",smax);
scanf("%i",&n);
if (n>smax) printf("Fora da faixa\n");
else
{
printf("Periodo %d \n",n);
p=pot(n);          // p = 2^n
for (i=0;i<=p-1;++i)    // Para i=0..2^n, converte i em um numero
{
    // binario de n digitos, e usa essa sequencia
    x.zera0();          // para indexar os 2^n pontos periodicos.
    x.um();              // Comeca com x=1.
    index[0]=0;
    for (j=1;j<=n;++j) index[j]=(((1 << (j-1)) & (i)) >> (j-1));
    for (j=n;j>=1;--j) printf("%u",index[j]); // Imprime a sequencia obtida.
    printf(" - ");
    k=0;
    do {
        // Loop de iteracao.
        k++;
        x0=x;
        for (j=1;j<=n;++j) {
            // Aplica as F0's e F1's na sequencia obtida.
            y=f(x,mi,index[j]);
            x=y;
        }
        } while((absol(x0-x)>eps)&&(k<=itmax)); // Enquanto a diferenca entre
        mostra(x); printf(" it. %d\n",k);      // iteracoes sucessivas e' maior
        x0.zera0();                            // que EPS e no. iteracoes<itmax.
    }
    // (EPS definido em FLUTPP.H)
}
}
}

```

```
// DENSEPP.CPP - Este programa calcula os primeiros 5 estagios da construcao
// de uma orbita densa do mapa quadratico, com um valor do parametro mi rece-
// bido do usuario.
```

```
#include <stdio.h>
#include "flutpp.h" // Ponto flutuante (o numero de digitos da mantissa e'
// definido em FLUTPP.H).
```

```
const smax=5; // Calcula ate' o passo smax.
```

```
static Ereal r0_5("0.5"),r0_25("0.25");
```

```
Ereal f(Ereal x, Ereal mi, int i) // Inversas locais.
```

```
{
Ereal y,b;
if (i==0) // F0(x)
{
x=r0_5-raiz(r0_25-x/mi);
}
else // F1(x)
{
x=r0_5+raiz(r0_25-x/mi);
}
return(x);
} // f
```

```
int pot(int n) // Retorna 2^n.
```

```
{
int a,i;
a=1;
if (n==0) return(a);
else
{
for (i=1;i<=n;++i) a=2*a;
return(a);
}
} // pot
```

```
void main() // Corpo principal.
```

```
{
int i,j,n,p;
Ereal mi,x,y;
int index[smax+1]; // Armazena a sequencia de 0's e 1's.

printf("Entre valor de mi (ate %d digitos signif.):",smax-3);
```

```

le_numero(&mi);
x=um;           // Inicia com x=1.
for(n=1;n<=smax;++n) {      // Vai do passo 1 ao smax.
    p=pot(n);           // p=2^n.
    for (i=0;i<=p-1;++i)    // Para i=0..2^n, converte i em um numero
    {                      // binario de n digitos.
        index[0]=0;
        for (j=1;j<=n;++j) index[j]=(((1 << (j-1)) & (i)) >> (j-1));
        printf(" ");
        for (j=n;j>=1;--j) printf("%u",index[j]); // Imprime a sequencia de digitos
        printf(" & ");           // binarios.
        {
            for (j=1;j<=n;++j) {          // Aplica a sequencia de F0's e
                y=f(x,mi,index[j]); // F1's correspondente `a sequen-
                x=y;           // cia de 0's e 1's obtida.
            }
        }
        mostra(x);           // Mostra o ponto da orbita
        printf("\n");        // obtido.
    }
}
}
}

```

```
/* FLUTPP.H - Arquivo header de definicao dos operadores de comparacao,  
operadores auxiliares de operadores aritmeticos para numeros do tipo Ereal .  
ETO 06/09/94.  
*/
```

```
#ifndef __FLUTPP_H  
#define __FLUTPP_H  
#endif
```

```
#include <iostream.h>
```

```
#if !defined( __DEFS_H )  
#include <_defs.h>  
#endif
```

```
#define TRUE 1  
#define FALSE 0
```

```
// defines usados para conversao para double.  
#define TAM_DOUBLE 25  
#define DIGITOS_DOUBLE 15
```

```
// Constantes importantes.  
#define ZERO "0"  
#define UM "1"  
#define EPS "1E-100"  
#define PI1_2 "1.5707963267948966192313216916397514420985"  
#define PI "3.141592653589793238462643383279502884197"  
#define PI3_2 "4.7123889803846898576939650749192543262955"  
#define PI_2 "6.283185307179586476925286766559005768394"  
#define E "2.7182818284590452353602874713526624977552"
```

```
// max_int = numero de digitos antes do ponto decimal (sempre 3)  
// max_dec = numero de digitos apos o ponto decimal (variavel)  
// max_tot = max_int+max_dec = numero total de digitos  
// max = max_tot+1  
#define max 104  
const max_int=3;  
const max_dec=100;  
const max_tot=103;
```

```
// Definicao da classe Ereal
```

```
_CLASSDEF(Ereal)
```

```
class _CLASSTYPE Ereal {
```

```

public:
// Operador =
Ereal& operator=(Ereal& x);

// Funcoes auxiliares
friend Ereal _FARFUNC subt_f(Ereal, Ereal);
friend Ereal _FARFUNC mult_alg(Ereal, char);
void desce_alg(char);
void desce_2alg(char, char);
int tam_int();
int tam_dec();
void desl_esq(int);
void desl_dir(int);
void zera();

// Construtores
Ereal();
Ereal(char *number);

// Destrutor
~Ereal();

// Conversao para double
friend double _FARFUNC dbl(Ereal _FAR &);
friend Ereal _FARFUNC re(double);

// Funcoes de E/S
friend void _FARFUNC mostra(Ereal _FAR &);
friend void _FARFUNC le_numero(Ereal *x);
friend ostream& operator<<(ostream&, Ereal _FAR &);
friend istream& operator>>(istream&, Ereal _FAR &);

// Operadores e funcoes
friend int _FARFUNC operator >(Ereal _FAR &, Ereal _FAR &);
friend int _FARFUNC operator <(Ereal _FAR &, Ereal _FAR &);
friend int _FARFUNC operator ==(Ereal _FAR &, Ereal _FAR &);
friend int _FARFUNC operator !=(Ereal _FAR &, Ereal _FAR &);
friend int _FARFUNC operator >=(Ereal _FAR &, Ereal _FAR &);
friend int _FARFUNC operator <=(Ereal _FAR &, Ereal _FAR &);
friend Ereal _FARFUNC operator +(Ereal _FAR &, Ereal _FAR &);
friend Ereal _FARFUNC operator -(Ereal _FAR &, Ereal _FAR &);
friend Ereal _FARFUNC operator *(Ereal _FAR &, Ereal _FAR &);
friend Ereal _FARFUNC operator /(Ereal _FAR &, Ereal _FAR &);
Ereal _FAR & operator+=(Ereal _FAR &);
Ereal _FAR & operator-=(Ereal _FAR &);

```



```

Ereal _FAR & operator*=(Ereal _FAR &);
Ereal _FAR & operator/=(Ereal _FAR &);
Ereal _FAR operator+();
Ereal _FAR operator-();

friend void _FARFUNC troca(Ereal _FAR &, Ereal _FAR &);
friend Ereal _FARFUNC r(char *);
friend Ereal _FARFUNC eleva(Ereal _FAR &, int);
friend Ereal _FARFUNC fatorial(int&);
friend Ereal _FARFUNC raiz(Ereal _FAR &);
friend Ereal _FARFUNC absol(Ereal _FAR &);
friend Ereal _FARFUNC frac(Ereal _FAR &);
friend Ereal _FARFUNC inteiro(Ereal _FAR &);
friend Ereal _FARFUNC exponencial(Ereal _FAR &);
friend Ereal _FARFUNC seno(Ereal &);
friend Ereal _FARFUNC cosseno(Ereal _FAR &);

private:
char d[max];
int e;
};

// As variaveis globais abaixo estao definidas em FLUTPP.CPP
extern Ereal zero,um,eps,pi1_2,pi,pi3_2,pi_2,e;

extern int operator >(Ereal _FAR &, Ereal _FAR &);
extern int operator <(Ereal _FAR &, Ereal _FAR &);
extern int operator ==(Ereal _FAR &, Ereal _FAR &);
extern int operator !=(Ereal _FAR &, Ereal _FAR &);
extern int operator >=(Ereal _FAR &, Ereal _FAR &);
extern int operator <=(Ereal _FAR &, Ereal _FAR &);

extern void troca(Ereal _FAR &, Ereal _FAR &);
extern Ereal r(char*);

extern Ereal operator +(Ereal _FAR &, Ereal _FAR &);
extern Ereal operator -(Ereal _FAR &, Ereal _FAR &);
extern Ereal operator *(Ereal _FAR &, Ereal _FAR &);
extern Ereal operator /(Ereal _FAR &, Ereal _FAR &);
extern Ereal eleva(Ereal _FAR &, int n);
extern Ereal fatorial(int&);

extern Ereal subt_f(Ereal, Ereal);
extern Ereal mult_alg(Ereal, char);

extern Ereal raiz(Ereal _FAR &);

```



```
extern Ereal absol(Ereal _FAR &);  
extern Ereal frac(Ereal _FAR &);  
extern Ereal inteiro(Ereal _FAR &);
```

```
extern void mostra(Ereal _FAR &);  
extern void le_numero(Ereal _FAR &);  
extern ostream& operator<<(ostream&, Ereal _FAR &);  
extern istream& operator>>(istream&, Ereal _FAR &);
```

```
extern double dbl(Ereal _FAR &);  
extern Ereal re(double);
```

```
extern Ereal exponencial(Ereal _FAR &);  
extern Ereal seno(Ereal _FAR &);  
extern Ereal cosseno(Ereal _FAR &);  
extern Ereal arctg(Ereal _FAR &);
```

```
/* FLUTPP.CPP - Arquivo que contem a implementacao das funcoes de
   FLUTPP.H , funcoes membro e amigas da classe Ereal.
   ETO 06/09/94
```

Observacoes sobre todas as operacoes aritmeticas e transcendentas :

1. No final de cada operacao, o numero e normalizado para que tenha exatamente 1 digito na parte inteira, caso nao seja igual a zero.
 2. Os resultados sao simplesmente truncados, isto e, nao sao arredondados
 3. Nao ha verificacao de overflow, underflow e perda de significancia
 4. Em caso de divisao por zero e erro de dominio, o programa e' terminado (sai para o sistema operacional).
- ```
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream.h>
#include "flutpp.h"
```

```
Ereal zero(ZERO),
 um(UM),
 eps(EPS),
 pi1_2(PI1_2),
 pi(PI),
 pi3_2(PI3_2),
 pi_2(PI_2),
 e(E);
```

```
// Zera o numero
inline void Ereal::zera()
{
 e=0;
 memset(d,0,max);
}
```

```
// Desloca n digitos para a esquerda (nao modifica o expoente)
inline void Ereal::desl_esq(int n)
{
 if (n<max_tot) {
 memmove(&(d[1]),&(d[n+1]),max_tot-n);
 memset(&(d[max_tot-n+1]),0,n);
 }
 else zera();
}
```

// Desloca n digitos para a direita (nao modifica o expoente)

inline void Ereal::desl_dir(int n)

```
{
if (n<=max_tot-1) {
memmove(&(d[n+1]),&(d[1]),max_tot-n);
memset(&(d[1]),0,n);
}
else zera();
}
```

// Construtor sem parametros - zera o numero

inline Ereal::Ereal()

```
{
e=0;
memset(d,0,max);
}
```

// Construtor com parametro string - numero passado no formato : d...d[.d...d][E[+/-]d...d]

Ereal::Ereal(char *number)

```
{
char a,h;
char *r;
int i,j,p;

e=0;
memset(d,0,max);

i=0; p=0;
d[0]=0;
if ((number[i]=='+') || (number[i]=='-'))
{
if (number[i]=='+') d[0]=1;
else d[0]=-1;
i++;
}
j=1; a='\0';
do {
a=number[i];
if ((a!='0') && (d[0]==0)) d[0]=1;
d[j]=a-'0';
j++; i++;
} while ((number[i]!='E') && (number[i]!='.') && (number[i]!='\0'));
if (d[1]!=0) p=j-2;
else p=-1;
h=FALSE;
if (number[i]=='.')

```

```

{
i++;
do {
a=number[i];
if ((a!='0') && (d[0]==0))
{
d[0]=1;
h=TRUE;
}
if ((a=='0') && (d[1]==0) && (!h)) p--;
d[j]=a-'0';
j++; i++;
} while ((number[i]!='E') && (number[i]!='\0'));
}
if (number[i]=='E')
{
r=strstr(number,"E");
r++;
e=atoi(r);
}
if (d[1]!=0) desl_dir(2);
else if (d[1]==0)
{
if (-p-2>0) desl_esq(-p-2);
else desl_dir(p+2);
}
e=e+p;
}

```

// Destrutor - nenhuma função

Ereal::~~Ereal()

```

{
}

```

// Operador de atribuição

inline Ereal& Ereal::operator=(Ereal& x)

```

{
e=x.e;
memmove(d,x.d,max);
return *this;
}

```

// Converte a string passada, no formato d...d[.d...d][E[+/-]d...d], para um Ereal

Ereal r(char *number)

```

{
char a,h;

```

```

char *s;
int i,j,p;
Ereal x;

x.zera();
i=0; p=0;
x.d[0]=0;
if ((number[i]=='+' || (number[i]=='-'))
{
 if (number[i]=='+') x.d[0]=1;
 else x.d[0]=-1;
 i++;
}
j=1; a='\0';
do {
 a=number[i];
 if ((a!='0') && (x.d[0]==0)) x.d[0]=1;
 x.d[j]=a-'0';
 j++; i++;
} while ((number[i]!='E') && (number[i]!='.') && (number[i]!='\0'));
if (x.d[1]!=0) p=j-2;
else p=-1;
h=FALSE;
if (number[i]!='.')
{
 i++;
 do {
 a=number[i];
 if ((a!='0') && (x.d[0]==0))
 {
 x.d[0]=1;
 h=TRUE;
 }
 if ((a=='0') && (x.d[1]==0) && (!h)) p--;
 x.d[j]=a-'0';
 j++; i++;
 } while ((number[i]!='E') && (number[i]!='\0'));
}
if (number[i]=='E')
{
 s=strstr(number,"E");
 s++;
 x.e=atoi(s);
}
if (x.d[1]!=0) x.desl_dir(2);
else if (x.d[1]==0)

```

```

{
if (-p-2>0) x.desl_esq(-p-2);
else x.desl_dir(p+2);
}
x.e=x.e+p;
return x;
}

```

// Retorna o numero de digitos da parte inteira do numero.

```

int Ereal::tam_int()
{
int i;
i=1;
while ((d[i]==0) && (i<=max_int)) i++;
return (max_int-i+1);
}

```

// Retorna o numero de digitos da parte decimal do numero

```

int Ereal::tam_dec()
{
int i;
i=max_tot;
while ((d[i]) && (i>max_int)) i--;
return(i-max_int);
}

```

// x = y ?

```

int operator==(Ereal& x, Ereal& y)
{
if (x.e!=y.e) return FALSE;
if (memcmp(x.d,y.d,max)!=0) return FALSE;
return TRUE;
}

```

// x <> y ?

```

int operator!=(Ereal& x, Ereal& y)
{
if (!(x==y)) return TRUE;
else return FALSE;
}

```

// x > y ?

```

int operator>(Ereal& x, Ereal& y)
{
int tx,ty;
int f,ig;

```



```

ig=(x==y);
if ((x.d[0]*y.d[0]<=0) && ((x.d[0]!=0) || (y.d[0]!=0)))
{
 if (x.d[0]<y.d[0]) f=FALSE;
 else f=TRUE;
}
else
{
 if (x.e<y.e) f=FALSE;
 else if (x.e>y.e) f=TRUE;
 else
 {
 f=TRUE;
 tx=x.tam_int();
 ty=y.tam_int();
 if (tx>ty) f=TRUE;
 else if (tx<ty) f=FALSE;
 else if ((tx==ty) && (!ig)) {
 if (memcmp(&(x.d[1]),&(y.d[1]),max-1)<0) f=FALSE;
 }
 else f=FALSE;
 }
 if ((x.d[0]<0) && (!ig)) f=!f;
}
return f;
}

```

```

// x < y ?
int operator<(Ereal& x, Ereal& y)
{
 if ((x>y) || (x==y)) return FALSE;
 else return TRUE;
}

```

```

// x >= y ?
int operator>=(Ereal& x, Ereal& y)
{
 if ((x>y) || (x==y)) return TRUE;
 else return FALSE;
}

```

```

// x <= y ?
int operator<=(Ereal& x, Ereal& y)
{
 if ((x<y) || (x==y)) return TRUE;
}

```

```

 else return FALSE;
 }

// Operador + unario
Ereal Ereal::operator+()
{
 return *this;
}

// Operador - unario
Ereal Ereal::operator-()
{
 this->d[0] = -this->d[0];
 return *this;
}

// numero += x
Ereal& Ereal::operator+=(Ereal& x)
{
 *this=*this+x;
 return *this;
}

// numero -= x
Ereal& Ereal::operator-=(Ereal& x)
{
 *this=*this-x;
 return *this;
}

// numero *= x
Ereal& Ereal::operator*=(Ereal& x)
{
 *this=(*this)*x;
 return *this;
}

// numero /= x
Ereal& Ereal::operator/=(Ereal& x)
{
 *this=(*this)/x;
 return *this;
}

// Troca x por y
void troca(Ereal& x, Ereal& y)

```

```

{
Ereal z;
z=y;
y=x;
x=z;
}

// Imprime o numero x em stdout.
void mostra(Ereal& x)
{
int i,j,k;
j=x.tam_dec();
k=x.tam_int();
if (x.d[0]==0) printf("0",stdout);
else {
if (x.d[0]<0) printf("-",stdout);
else if (x.d[0]>0) printf("+",stdout);
for (i=max_int-k+1;i<=max_int;++i) printf("%u",x.d[i],stdout);
if (j!=0) printf(".",stdout);
for (i=max_int+1;i<=max_int+j;++i) printf("%u",x.d[i],stdout);
if (x.d[0]==0) printf("0",stdout);
printf("E",stdout);
printf("%+i",x.e,stdout);
}
}

// Le um numero de stdin
void le_numero(Ereal *x)
{
char *k;
char a,h;
int i,j,p;

if ((k=new char[max+9])==NULL) exit;
(*x).zera();
k[0]='\0';
gets(k);
i=0; p=0;
x->d[0]=0;
if ((k[i]=='+') || (k[i]=='-'))
{
if (k[i]=='+') x->d[0]=1;
else x->d[0]=-1;
i++;
}
j=1; a='\0';

```

```

do {
 a=k[i];
 if ((a!='0') && (x->d[0]==0)) x->d[0]=1;
 x->d[j]=a-'0';
 j++; i++;
 } while ((k[i]!='E') && (k[i]!='.') && (k[i]!='\0'));
if (x->d[1]!=0) p=j-2;
else p=-1;
h=FALSE;
if (k[i]!='.')
{
 i++;
 do {
 a=k[i];
 if ((a!='0') && (x->d[0]==0))
 {
 x->d[0]=1;
 h=TRUE;
 }
 if ((a=='0') && (x->d[1]==0) && (!h)) p--;
 x->d[j]=a-'0';
 j++; i++;
 } while ((k[i]!='E') && (k[i]!='\0'));
 }
if (k[i]=='E')
{
 i++;
 x->e=atoi(&(k[i]));
 }
if (x->d[1]!=0) (*x).desl_dir(2);
else if (x->d[1]==0)
{
 if (-p-2>0) (*x).desl_esq(-p-2);
 else (*x).desl_dir(p+2);
 }
x->e=x->e+p;
delete [] k;
}

// Operador de insercao
ostream& operator<<(ostream& stream, Ereal& x)
{
 mostra(x);
 return stream;
}

```

```

// Operador de extracao
istream& operator>>(istream& stream, Ereal& x)
{
 le_numero(&x);
 return stream;
}

// Desloca 1 digito para esquerda e insere um digito k na ultima posicao
inline void Ereal::desce_alg(char k)
{
 if ((d[0]==0) && (k!=0)) d[0]=1;
 desl_esq(1);
 d[max_tot]=k;
}

// Desloca 2 digitos para esquerda e insere os digitos k,j nas ultimas posicoes
inline void Ereal::desce_2alg(char k ,char j)
{
 if ((d[0]==0) && ((k!=0) || (j!=0))) d[0]=1;
 desl_esq(2);
 d[max_tot-1]=k;
 d[max_tot]=j;
}

Ereal operator-(Ereal& x, Ereal& y);

// Operador + binario
Ereal operator+(Ereal& x, Ereal& y)
{
 Ereal u,v,z;
 int i;
 char r,s;
 int k,t;
 char sinalx,sinaly;

 z.zera();
 u.zera();
 v.zera();
 u=x;
 v=y;

 if (u.d[0]*v.d[0]<0)
 {
 sinalx=u.d[0];
 sinaly=v.d[0];
 u.d[0]=1;
 }

```



```

v.d[0]=1;
if (u>v)
{
z=u-v;
z.d[0]=sinalx;
}
else if (u<v)
{
z=v-u;
z.d[0]=sinaly;
}
}
else if (u.d[0]*v.d[0]>0)
{
if (u.e>v.e) {
k=u.e-v.e;
z.e=u.e;
v.desl_dir(k);
}
else if (u.e<v.e)
{
k=v.e-u.e;
z.e=v.e;
u.desl_dir(k);
}
else z.e=u.e;
if ((u.d[0]==0) && (v.d[0]==0)) z.d[0]=0;
else if (u.d[0]==0) z.d[0]=v.d[0];
else if (v.d[0]==0) z.d[0]=u.d[0];
else z.d[0]=u.d[0];
r=0;
k=u.tam_dec();
t=v.tam_dec();
if (k>t) i=max_int+k;
else i=max_int+t;
k=u.tam_int();
t=v.tam_int();
if (t>k) k=t;
while (i>=max_int-k)
{
s=u.d[i]+v.d[i]+r;
z.d[i]=s % 10;
r= s / 10;
i--;
}
}

```

```

else {
 if ((v.d[0]==0) && (u.d[0]==0)) z.zera();
 else if (u.d[0]==0) z=v;
 else if (v.d[0]==0) z=u;
}

if (z.d[0]!=0)
{
 if (z.tam_int()>1) do
 {
 z.desl_dir(1);
 z.e=z.e+1;
 } while (z.tam_int()>1);
 else if (z.tam_int()<1) do
 {
 z.desl_esq(1);
 z.e=z.e-1;
 } while (z.tam_int()<1);
}
return(z);
}

// Operador - binario
Ereal operator-(Ereal& x, Ereal& y)
{
 int i;
 char r,s;
 int k,t;
 int sinal;
 Ereal u,v,z;

 u.zera();
 v.zera();
 u=x;
 v=y;
 z.zera();

 if (u.d[0]*v.d[0]<=0)
 {
 if ((u.d[0]==0) && (v.d[0]==0)) z.d[0]=0;
 else if (u.d[0]==0)
 {
 z=v;
 z.d[0]=-v.d[0];
 }
 else if (v.d[0]==0) z=u;
 }

```

```

else
{
 sinal=u.d[0];
 u.d[0]=1;
 v.d[0]=1;
 z=u+v;
 if (sinal>0) z.d[0]=1;
 else z.d[0]=-1;
}
}
else
{
 if (u.e>v.e) {
 k=u.e-v.e;
 z.e=u.e;
 v.desl_dir(k);
 }
 else if (u.e<v.e)
 {
 k=v.e-u.e;
 z.e=v.e;
 u.desl_dir(k);
 }
 else z.e=u.e;
 sinal=u.d[0];
 u.d[0]=1;
 v.d[0]=1;
 if (u>v) z.d[0]=sinal;
 else if (u<v)
 {
 z.d[0]=-sinal;
 troca(u,v);
 }
 else if (u==v) z.d[0]=0;
 r=0;
 k=u.tam_dec();
 t=v.tam_dec();
 if (k>t) i=max_int+k;
 else i=max_int+t;
 k=u.tam_int();
 t=v.tam_int();
 if (t>k) k=t;
 while (i>=max_int-k)
 {
 s=u.d[i]-v.d[i]-r;
 z.d[i]=(10+s) % 10;
 }
}

```

```

 r= ! ((10+s) / 10);
 i--;
 }
}
if (z.d[0]!=0)
{
 if (z.tam_int()>1) do
 {
 z.desl_dir(1);
 z.e=z.e+1;
 } while (z.tam_int()>1);
 else if (z.tam_int()<1) do
 {
 z.desl_esq(1);
 z.e=z.e-1;
 } while (z.tam_int()<1);
}
return(z);
}

```

// Subtrai dois numeros e nao normaliza o resultado (isto e, o resultado nao esta necessariamente na forma d.d...dE+d...d

Ereal subt_f(Ereal x, Ereal y)

```

{
 int i;
 char r,s;
 int k,t;
 int sinal;
 Ereal z;

 z.zera();
 if (x.e>y.e) {
 k=x.e-y.e;
 z.e=x.e;
 y.desl_dir(k);
 }
 else if (x.e<y.e)
 {
 k=y.e-x.e;
 z.e=y.e;
 x.desl_dir(k);
 }

 if (x.d[0]*y.d[0]<=0)
 {
 if ((x.d[0]==0) && (x.d[0]==0)) z.d[0]=0;

```

```

else if (x.d[0]==0)
{
 z=y;
 z.d[0]=-y.d[0];
}
else if (y.d[0]==0) z=x;
else
{
 sinal=x.d[0];
 x.d[0]=1;
 y.d[0]=1;
 z=x+y;
 if (sinal>0) z.d[0]=1;
 else z.d[0]=-1;
}
}
else
{
 sinal=x.d[0];
 x.d[0]=1;
 y.d[0]=1;
 if (x>y) z.d[0]=sinal;
 else if (x<y)
 {
 z.d[0]=-sinal;
 troca(x,y);
 }
 else if (x==y) z.d[0]=0;
 r=0;
 k=x.tam_dec();
 t=y.tam_dec();
 if (k>t) i=max_int+k;
 else i=max_int+t;
 k=x.tam_int();
 t=y.tam_int();
 if (t>k) k=t;
 while (i>=max_int-k)
 {
 s=x.d[i]-y.d[i]-r;
 z.d[i]=(10+s) % 10;
 r=((10+s) / 10);
 i--;
 }
}
return(z);
}

```



```

// Operador *
Ereal operator*(Ereal& x, Ereal& y)
{
 int i,j,k;
 int ix,dx,iy;
 char r,r1,s,s1;
 char *temp;
 char *temp1;
 Ereal u,v,z;

 if ((temp=new char[2*max_tot+1])==NULL) {
 printf("Sem memoria\n"); exit(1); }
 if ((temp1=new char[2*max_tot+1])==NULL) {
 printf("Sem memoria\n"); exit(1); }

 u.zera();
 v.zera();
 z.zera();
 if ((x.d[0]==0) || (y.d[0]==0)) return(z);
 u=x;
 v=y;
 for (i=0;i<=2*max_tot;++i)
 {
 temp[i]=0;
 temp1[i]=0;
 }
 z.d[0]=u.d[0]*v.d[0];
 z.e=u.e+v.e;
 i=max_int+v.tam_dec();
 iy=v.tam_int();
 dx=u.tam_dec();
 ix=u.tam_int();
 while (i>=max_int-iy+1)
 {
 r=0; s=0; r1=0;
 j=max_int+dx;
 while (j>=max_int-ix)
 {
 s=v.d[i]*u.d[j]+r;
 temp[i+j]=s % 10;
 r= s / 10;
 s1=temp[i+j]+temp1[i+j]+r1;
 temp[i+j]=s1 % 10;
 r1= s1 / 10;
 }
 i--;
 }
 z.tam_dec=i;
 z.tam_int=i;
 return(z);
}

```

```

j--;
}
temp[i+j]=temp[i+j]+temp1[i+j]+r1;
for (k=1;k<=max_int+dx+i;++k)
{
 temp1[k]=temp[k];
 temp[k]=0;
}

i--;
}
memcpy(&(z.d[1]),&(temp1[max_int+1]),max_tot);
if (z.d[0]!=0)
{
 if (z.tam_int()>1) do
 {
 z.desl_dir(1);
 z.e=z.e+1;
 } while (z.tam_int()>1);
 else if (z.tam_int()<1) do
 {
 z.desl_esq(1);
 z.e=z.e-1;
 } while (z.tam_int()<1);
}
delete[] temp;
delete[] temp1;
return(z);
}

```

```

// Multiplica x pelo digito k
Ereal mult_alg(Ereal x, char k)
{
 Ereal z;

```

```

 if (k==0) z.zera();
 else
 {
 int i,t;
 char r,s;
 z.zera();
 if (k*x.d[0]<0) z.d[0]=-1;
 else if (k*x.d[0]>0) z.d[0]=1;
 else z.d[0]=0;
 i=max_int+x.tam_dec();
 r=0; s=0;
 t=x.tam_int();

```

```

while (i>=max_int-t)
{
s=x.d[i]*k+r;
z.d[i]=s % 10;
r= s / 10;
i--;
}
}
return(z);
}

```

```

// Operador / (divisao por tentativas)
Ereal operator/(Ereal& a, Ereal& b)

```

```

{
int i,j;
int l,m;
char k;
Ereal x,y,r,s,t,u;
char chave;
int ponto;

if (b.d[0]==0)
{
printf("\nDivisao por zero.\n");
exit(1);
}
r.zera();
s.zera();
t.zera();
if (a.d[0]==0) return(r);
r.d[0]=a.d[0]*b.d[0];
r.e=a.e-b.e;
x=a;
y=b;
x.d[0]=1;
y.d[0]=1;
x.e=0;
y.e=0;
chave=FALSE;
ponto=0;

if (x==y) {
r.d[max_int]=1;
ponto=1;
}
else if (x>y)

```

```

{
i=1;
while (t<y)
{
t.desl_esq(1);
t.d[max_tot]=x.d[max_int-x.tam_int()+i];
t.d[0]=1;
i++;
}
l=x.tam_int()-t.tam_int();
i=1; j=1;
chave=FALSE;
do
{
k=10;
do
{
k--;
s.zera();
s=mult_alg(y,k);
} while (s>t);
if ((ponto==0) && (j>1)) ponto=i;
if ((k==0) && (chave==TRUE))
{
if (max_int-1+j<=max_tot) k=x.d[max_int-1+j];
else k=0;

t.desce_alg(k);
j++;
r.d[i]=0;
i++;
chave=TRUE;
}
else if ((k==0) && (chave==FALSE))
{
chave=TRUE;
if (max_int-1+j<=max_tot) k=x.d[max_int-1+j];
else k=0;

t.desce_alg(k);
j++;
}
else {
r.d[i]=k;
i++;
u=subt_f(t,s);
t=u;
chave=FALSE;
}
}

```

```

 }
} while (i<=max_tot);
r.desl_dir(max_int-ponto);
}
else if ((x<y) && (x.tam_int()+x.tam_dec()!=0))
{
i=0;
t=x;
while (t<y)
{
i++;
t.desl_esq(1);
}
do
{
k=10;
do
{
k--;
s.zera();
s=mult_alg(y,k);
} while (s>t);
if (k==0)
{
t.desce_alg(0);
if (chave==TRUE) {
r.d[i]=0;
i++;
}

chave=TRUE;
}
}
else
{
chave=FALSE;
r.d[i]=k;
i++;
u=subt_f(t,s);
t=u;
}
} while (i<=max_tot);
r.desl_dir(max_int);
}
if (r.d[0]!=0)
{
if (r.tam_int()>1) do
{

```



```

 r.desl_dir(1);
 r.e=r.e+1;
 } while (r.tam_int()>1);
else if (r.tam_int()<1) do
 {
 r.desl_esq(1);
 r.e=r.e-1;
 } while (r.tam_int()<1);
}
return(r);
}

// Raiz quadrada (por tentativas)
Ereal raiz(Ereal& x)
{
 int i,j;
 char k;
 Ereal r,s,t,u;

 if (x.d[0]<0)
 {
 printf("\nErro de dominio.\n");
 exit(1);
 }
 r.zera();
 if (x.d[0]!=0)
 {
 if (((x.e) % 2)!=0)
 {
 x.desl_esq(1);
 x.e=x.e-1;
 }
 r.e=(x.e)/2;
 s.zera();
 t.zera();
 u.zera();
 i=max_int-1;
 j=1;
 do
 {
 if ((i+1)<max_tot) t.desce_2alg(x.d[i],x.d[i+1]);
 else t.desl_esq(2);

 k=10;
 do
 {
 s=r;

```

```

u=mult_alg(s,2);
s=u;
k--;
s.desce_alg(k);
u=mult_alg(s,k);
s=u;
u.zera();
} while (s>t);
r.desce_alg(k);
u=subt_f(t,s);
t=u;
i=i+2;
j++;
} while (j<=max_dec);
r.desl_esq(1);
if (r.d[0]!=0)
{
if (r.tam_int()>1) do
{
r.desl_dir(1);
r.e=r.e+1;
} while (r.tam_int()>1);
else if (r.tam_int()<1) do
{
r.desl_esq(1);
r.e=r.e-1;
} while (r.tam_int()<1);
}
}
return(r);
}

```

```

// Valor absoluto
Ereal absol(Ereal& x)
{
Ereal y;

y.zera();
y=x;
if (x.d[0]<0)
{
y.d[0]=-x.d[0];
}
return(y);
}

```

```

// Parte inteira
Ereal inteiro(Ereal& x)
{
 int i,k;
 Ereal y;

 y.zera();
 k=x.e;
 if (k>=0)
 {
 y=x;
 for (i=max_int+k+1;i<=max_tot;++i) y.d[i]=0;
 }
 return(y);
}

```

```

// Parte fracionaria
Ereal frac(Ereal& x)
{
 int i,k;
 Ereal y;

 y.zera();
 k=x.e;
 y=x;
 if (k>=0)
 {
 if ((k>=max_dec) || (k>=y.tam_dec())) y.zera();
 else
 {
 for (i=1;i<=max_int+k;++i) y.d[i]=0;
 y.desl_esq(k+1);
 y.e=y.e-k-1;
 }
 }
 return(y);
}

```

```

// Eleva x ao inteiro n
Ereal eleva(Ereal& x, int n)
{
 int i;
 Ereal y,z;

 z.zera();
 z.d[0]=1;

```

```

z.d[max_int]=1;
if (n!=0)
{
for (i=1;i<=n;++i) {
 y=z*x;
 z=y;
 }
return(z);
}

// Converte x para double
double dbl(Ereal& x)
{
double y,z;
int i,e,sinal;

y=0; sinal=0; e=0;
if (x.d[0]!=0)
{
sinal=x.d[0];
y=1; z=0;
for (i=max_int;i>=1;--i)
{
z=z+x.d[i]*y;
y=y*10;
}
y=10;
for (i=1;i<=max_dec;++i)
{
z=z+x.d[max_int+i]/y;
y=y*10;
}
e=x.e;
if (e>0) {
for (i=1;i<=e;++i) z=z*10;
}
else if (e<0) {
e=-e;
for (i=1;i<=e;++i) z=z/10;
}
}
z=z*sinal;
return(z);
}

```

```
// Converte x para Ereal
Ereal re(double x)
{
 Ereal y;
 char buf[TAM_DOUBLE];

 gcvt(x,DIGITOS_DOUBLE,buf);
 strupr(buf);
 y=r(buf);
 return(y);
}
```

```
// Exponencial
Ereal exponencial(Ereal& x)
{
```

```
 Ereal m,n,y,z,xn,w;
```

```
 n.zera();
 n=um;
 z.zera();
 y.zera();
 y=um;
 xn.zera();
 xn=um;
 w.zera();
 m.zera();
 do
 {
 w=z+y;
 z=w;
 w=xn*x;
 xn=w;
 w=m+um;
 m=w;
 w=n*m;
 n=w;
 y=xn/n;
 } while (absol(y)>eps);
 return(z);
}
```

```
// Seno
Ereal seno(Ereal& x)
{
 int i,snl;
```

```

Ereal m,n,y,z,xn,x2,w;

y.zera();
y=x;
snl=0;
if (y.d[0]<0) {
 snl=-1;
 y.d[0]=-y.d[0];
}
else if (y.d[0]>0) snl=1;

if (y>=pi_2)
{
 Ereal v;

 v.zera();
 v=y/pi_2;
 v=inteiro(v);
 v=v*pi_2;
 y=y-v;
}

if ((y>=pi1_2) && (y<pi))
{
 y=pi-y;
}
else if ((y>=pi) && (y<pi3_2))
{
 y=y-pi;
 snl=-snl;
}
else if ((y>=pi3_2) && (y<pi_2))
{
 y=pi_2-y;
 snl=-snl;
}

m.zera();
m=um;
n.zera();
n=um;
z.zera();
x2.zera();
x2=eleva(y,2);
xn.zera();
xn=y;

```



```

w.zera();

i=0;
do
{
if ((i % 2) == 0) {
 w=z+y;
 z=w;
 }

 else {
 w=z-y;
 z=w;
 }
 w=xn*x2;
 xn=w;
 i++;
 w=m+um;
 m=w;
 w=n*m;
 n=w;
 w=m+um;
 m=w;
 w=n*m;
 n=w;
 y=xn/n;
} while (y>eps);
z.d[0]=snl*z.d[0];
return(z);
}

```

```

// Cosseno
Ereal cosseno(Ereal& x)
{
 int i,snl;
 Ereal m,n,y,z,xn,x2,w;

```

```

y.zera();
y=x;
snl=1;
if (y.d[0]<0) {
 y.d[0]=-y.d[0];
 }

```

```

if (y>=pi_2)
{
 Ereal v;

```

```

v.zera();
v=y/pi_2;
v=inteiro(v);
v=v*pi_2;
y=y-v;
}

if ((y>=pi1_2) && (y<pi))
{
y=pi-y;
snl=-snl;
}
else if ((y>=pi) && (y<pi3_2))
{
y=y-pi;
snl=-snl;
}
else if ((y>=pi3_2) && (y<pi_2))
{
y=pi_2-y;
}

```

```

n.zera();
n=um;
z.zera();
x2.zera();
x2=eleva(y,2);
y.zera();
y=um;
xn.zera();
xn=um;
w.zera();
m.zera();
i=0;
do
{
if ((i % 2) == 0) {
w=z+y;
z=w;
}

else {
w=z-y;
z=w;
}
}
w=xn*x2;

```

```

xn=w;
i++;
w=m+um;
m=w;
w=n*m;
n=w;
w=m+um;
m=w;
w=n*m;
n=w;
y=xn/n;
} while (y>eps);
z.d[0]=snl*z.d[0];
return(z);
}

```

// Arco-tangente

Ereal arctg(Ereal& x)

```

{
int i,snl;
Ereal n,y,z,xn,x2,dois("2");

```

```

if(x>=r("1") && x<=r("1")) {

```

```

 xn=x;
 x2=-eleva(x,2);
 n=um;
 z.zera();
 do {
 y = xn/n;
 z += y;
 xn *= x2;
 n += dois;
 } while(absol(y)>eps);
}

```

```

else {
 xn=um/x;
 x2=-eleva(x,2);
 n=um;
 z.zera();
 do {
 y = xn/n;
 z += y;
 xn *= x2;
 n += dois;
 } while(absol(y)>eps);
 z = pi1_2-z;
}

```

```
}
return(z);
}
```

// Fatorial do inteiro n

Ereal fatorial(int& n)

```
{
int i;
Ereal k,r,x;
x.zera();
k.zera();
r.zera();
x=um;
if (n!=0)
{
for (i=1;i<=n;++i)
{
r=k+um;
k=r;
r=x*k;
x=r;
}
}
return x;
}
```

(* Este programa feito em Mathematica calcula o ponto periodico de *)
 (* periodo 20 (e sua orbita) que e' mostrado na tabela D.4. *)
 (* Observe que os calculos sao feitos com precisao de 60 digitos *)
 (* significativos. *)

```
(* Funcao logistica *)
mi:=SetPrecision[4.01,60]
F[x_]:=mi*SetPrecision[x,60]*(1-SetPrecision[x,60])
```

```
(* Inversas *)
f[x_]:=SetPrecision[0.5,60]-Sqrt[SetPrecision[0.25,60]-SetPrecision[x,60]/mi]
g[x_]:=SetPrecision[0.5,60]+Sqrt[SetPrecision[0.25,60]-SetPrecision[x,60]/mi]
```

(* Composta *)
h[x]:=f[f[f[f[f[f[f[f[f[g[f]g[g[g[x]]]]]]]]]]]

```
(* Loop para calculo, 20 iteracoes; o resultado final esta na variavel x *)
For[x=1; i=1, i<=20, i++, x=SetPrecision[h[x],60]]
```

(* Fim *)