ANÁLISE E SIMULAÇÃO DE DETRITOS ESPACIAIS

RELATÓRIO FINAL DE PROJETO DE INCIAÇÃO CIENTÍFICA
(PIBIC/CNPq/INPE)

Bolsista – Andreza da Costa Batista (ETEP Bolsista PIBIC/CNPq)

E-mail: andreza999@itelefonica.com.br

Orientador - Dr. Marcelo Lopes de Oliveira e Souza (DMC/ETE/INPE)
E-mail: marcelo@dem.inpe.br

Julho de 2007

SUMÁRIO **página**

LISTA DE FIGURAS E TABELAS **página**

CAPÍTULO 1 – INTRODUÇÃO

BREVE HISTÓRICO: No período de fevereiro a julho de 2006 foram realizados: 1) Nossa introdução ao tema Detritos Espaciais; 2) Uma análise de casos recentes; 3) O estudo de três formulações matemáticas para a propagação dos detritos, a saber: 3.1) Um 1º Método Numérico, utilizando o programa KK e suas adaptações para simular a propagação da distribuição inicial de um conjunto de N partículas iguais resultantes da explosão radial com gradiente β de um satélite circular e homogêneo com velocidades de translação e rotação; 3.2) Um 1º Modelo Analítico (Geométrico) utilizando um programa em C com auxílio do MS Visual C++ 6.0 do ambiente MS Visual Studio 6.0; os parâmetros utilizados nesse processo foram o tempo e as coordenadas do centro de atração gravitacional; assumiu-se que cada detrito tinha a mesma velocidade angular constante; 3.3) Um 2º Modelo Analítico (Geométrico) com o qual foram feitas simulações utilizando o MATLAB, considerando a nuvem de detritos espaciais uma elipse deformada.

Os resultados e conclusões deste período estão detalhados no Relatório Final de Projeto de Iniciação Científica: Análise e Simulação de Detritos Espaciais – Julho de 2006, que foi apresentado no SINCIPE 2006.

OBJETIVOS E ESTRUTURA DESTE RELATÓRIO: Este Relatório Final objetiva apresentar os resultados obtidos no período de agosto de 2006 a julho de 2007, que incluem: 1) Um Estudo de Mapeamentos Complexos para o 2º Modelo Analítico, começando com um quadrado com ângulos retos em $\pm 1.5 \pm 1.5i$, utilizando o MATLAB, detalhado no Capítulo 2, e cujos programas estão no Apêndice A; 2) Um Estudo da Curvatura dos Eixos Maiores da Elipse para o 2º Modelo Analítico (Geométrico), utilizando o MATLAB, detalhado no Capítulo 3, e cujos programas estão no Apêndice B; 3) Uma Aproximação do Bananóide pela Elipse Curvada com o Método Iterativo, detalhado no Capítulo 4, e cujos programas estão no Apêndice B; 4) Um Ajuste de uma Elipse a um Conjunto de Pontos pelo Método dos Mínimos Quadrados detalhado no Capítulo 5, e cujos programas estão detalhados no Apêndice B; 5) Um Estudo de Métodos de Integração Numérica para o 1º Modelo Numérico, detalhado no Apêndice C; 6) Um Estudo do MATLAB para todos os Modelos, detalhados no Apêndice D.

CAPÍTULO 2 – ESTUDO DE MAPEAMENTOS COMPLEXOS PARA O 2º MODELO ANALÍTICO (GEOMÉTRICO)

Os mapeamentos complexos apresentam muitas propriedades interessantes e passíveis de serem usadas no 2º Modelo Analítico para aproximar os vários efeitos observados no 2º Método Numérico. Estes efeitos transformam sucessivamente um círculo numa elipse, bananóide, amebóide, etc., a saber: translação, rotação, crescimento, flexão, torcimento, etc. Para estudá-las, iniciamos com alguns exemplos:

EXEMPLO 1: Considere a função $f(z) = z^2$. Nós a analisaremos, utilizando o MATLAB (vide Apêndice A, Programas 6 e 7), para ver como um quadrado com ângulos retos em $\pm 1.5 \pm 1.5i$ será mapeado. Primeiramente olharemos apenas para metade do quadrado localizado no semi-plano direito, $f(z) = z$ (Figura 2.1). Esta região foi mapeada uma por uma, ou seja, quadramos cada segmento de reta representado no quadrado, e obtivemos várias parábolas (Figura 2.2).

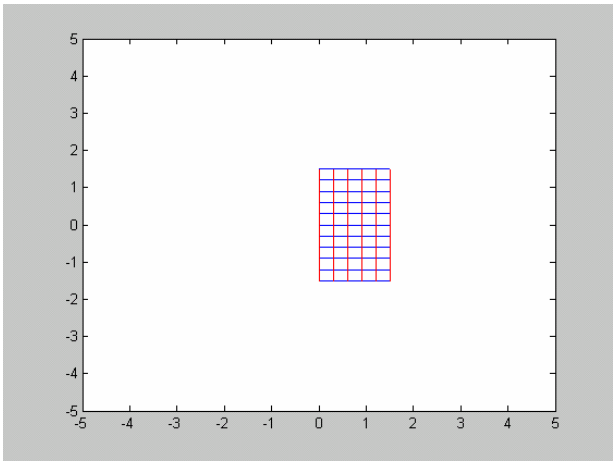

Figura 2.1 – $f(z) = z$

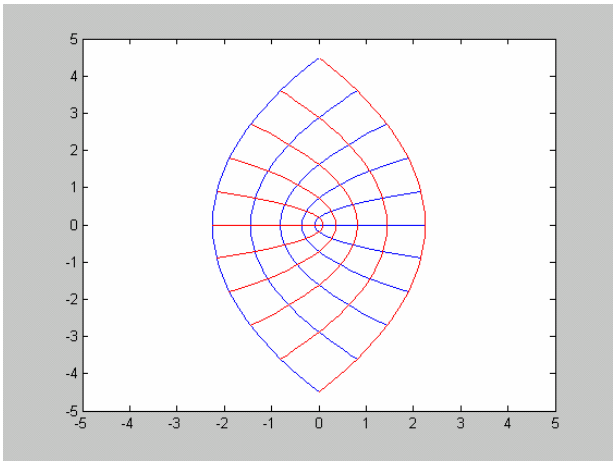

Figura 2.2 – $f(z) = z^2$

Podemos observar que os segmentos de reta com constantes reais foram mapeadas em parábolas com concavidade voltada para o eixo real negativo; e os segmentos de reta que possuem constantes imaginárias foram mapeadas em parábolas com concavidade voltada para o eixo real positivo. Notamos também que as parábolas se encontram nos pontos que possuem ângulos retos no quadrado.

EXEMPLO 2: Se fizermos $f(z)=(1-t)z$, com t variando de 0 a 1 (vide apêndice A, Programas 8 a 18), com um incremento de 0.1 teremos (Figuras 2.3 a 2.13):

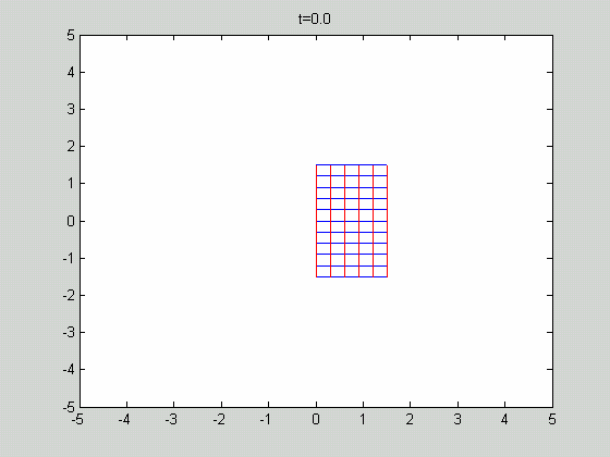

Figura 2.3 – $f(z) = (1-t)z$ para $t = 0.0$

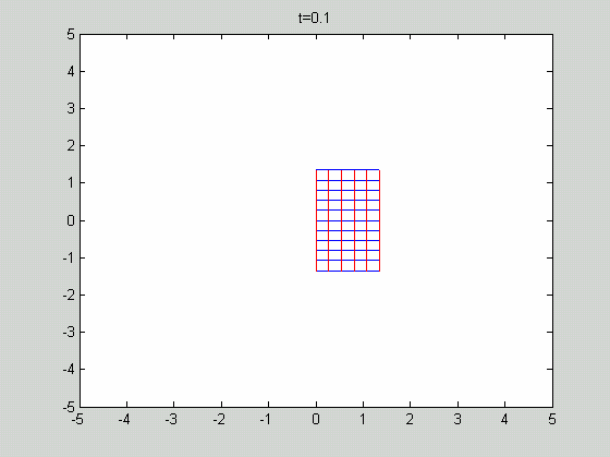

Figura 2.4 – $f(z) = (1-t)z$ para $t = 0.1$

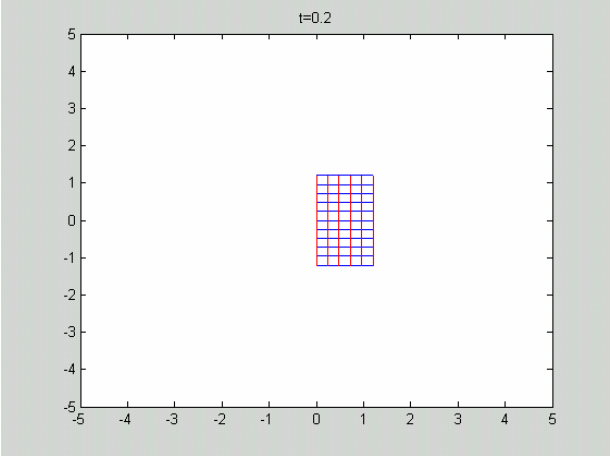

Figura 2.5 – $f(z) = (1-t)z$ para $t = 0.2$

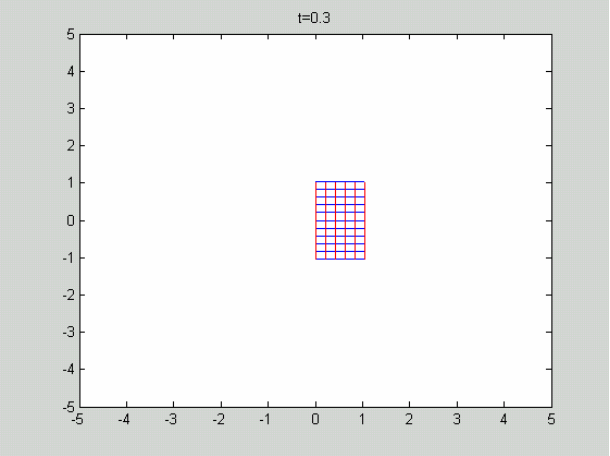

Figura 2.6 – $f(z) = (1-t)z$ para $t = 0.3$

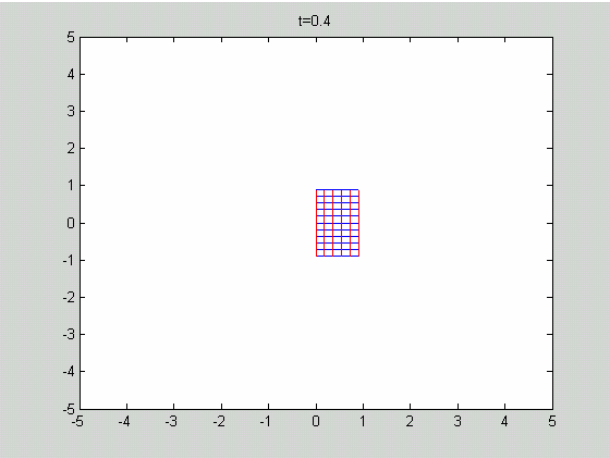

Figura 2.7 – $f(z) = (1 - t)z$ para $t = 0.4$

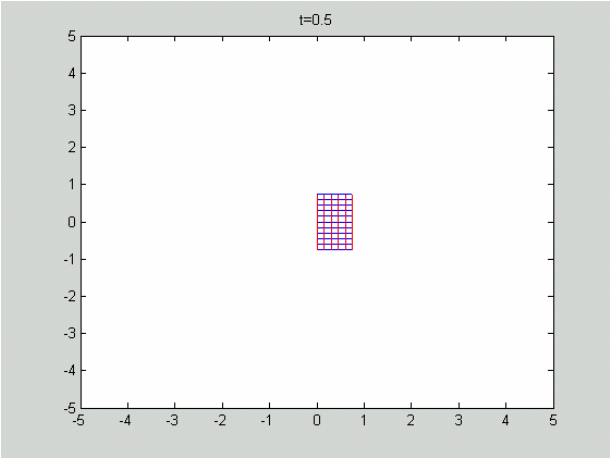

Figura 2.8 – $f(z) = (1 - t)z$ para $t = 0.5$

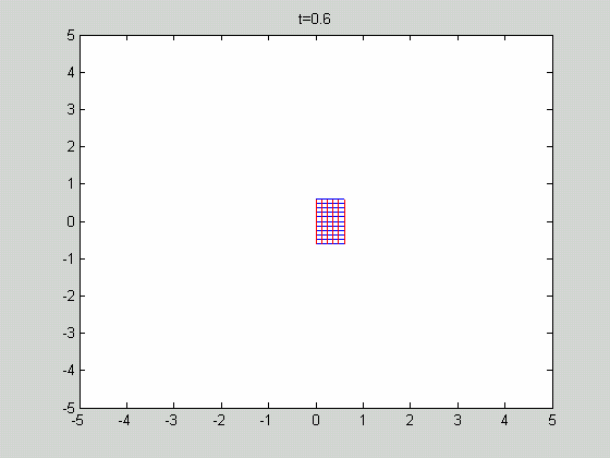

Figura 2.9 – $f(z) = (1 - t)z$ para $t = 0.6$

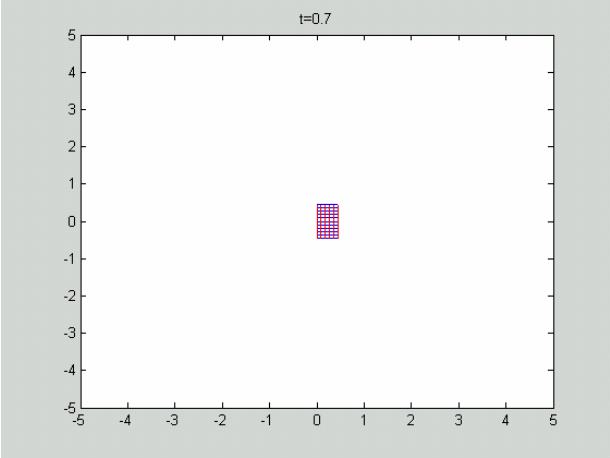

Figura 2.10 – $f(z) = (1 - t)z$ para $t = 0.7$

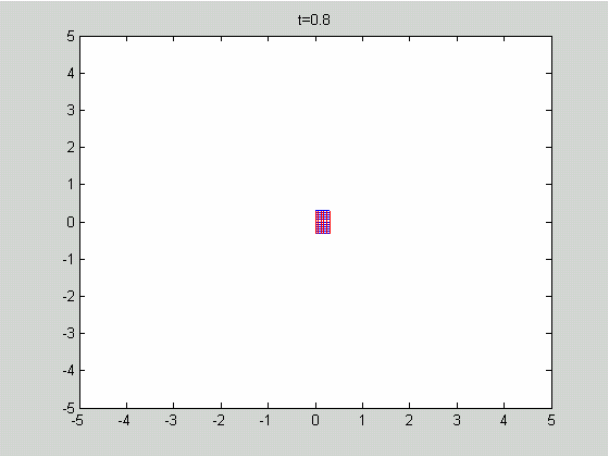

Figura 2.11 – $f(z) = (1 - t)z$ para $t = 0.8$

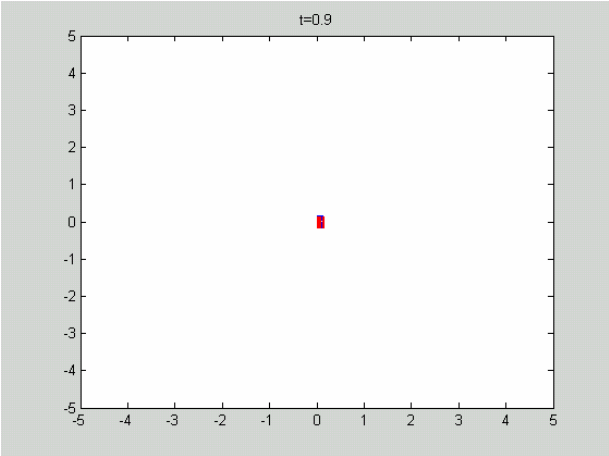

Figura 2.12 – $f(z) = (1 - t)z$ para $t = 0.9$

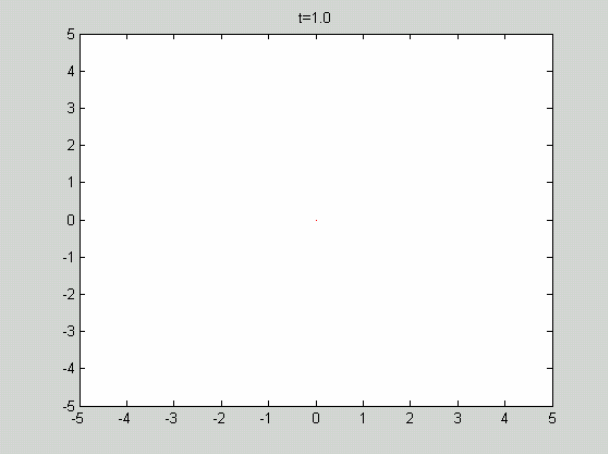

Figura 2.13 – $f(z) = (1 - t)z$ para $t = 1.0$

Podemos notar que a medida que t aumenta, a metade do quadrado localizado no semi-plano direito diminui até virar um ponto (0,0).

EXEMPLO 3: Se fizermos $f(z)=tz^2$, com t variando de 0 a 1 (vide apêndice A, Programas 19 a 29), com um incremento de 0.1 teremos (Figuras 2.14 a 2.24):

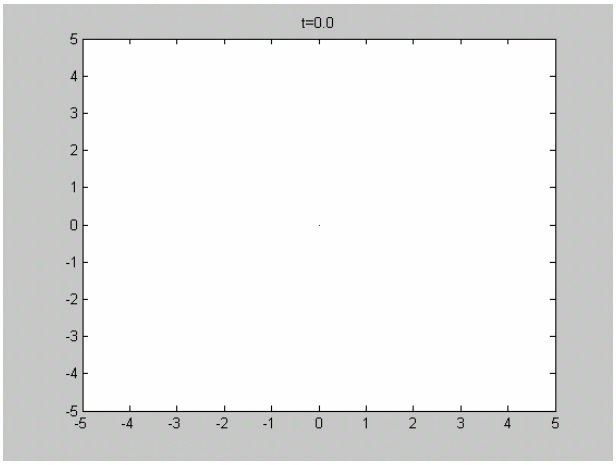

Figura 2.14 – $f(z) = tz^2$ para $t = 0.0$

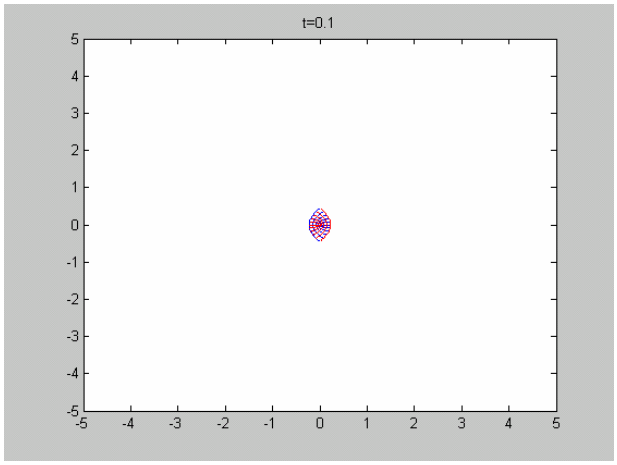

Figura 2.15 – $f(z) = tz^2$ para $t = 0.1$

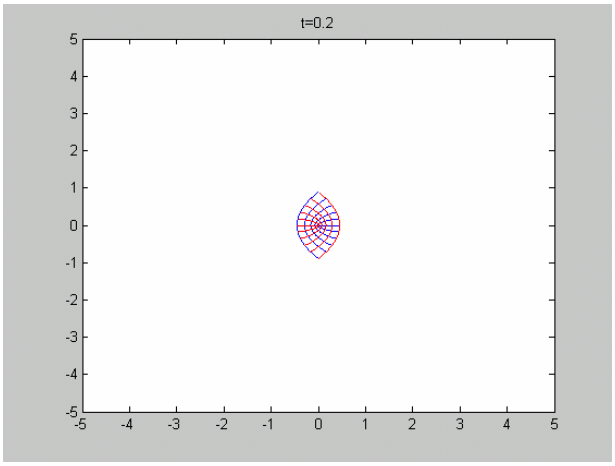

Figura 2.16 – $f(z) = tz^2$ para $t = 0.2$

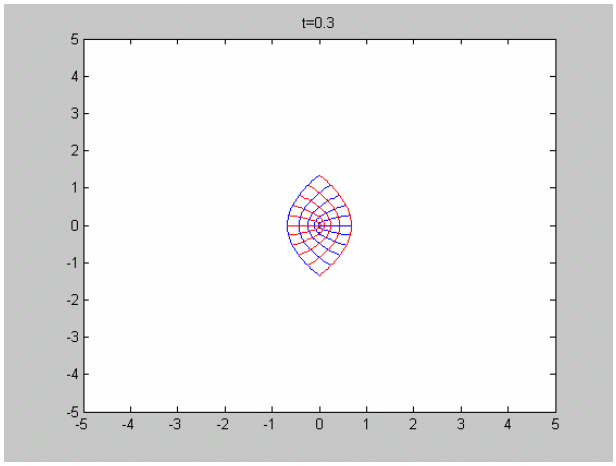

Figura 2.17 – $f(z) = tz^2$ para $t = 0.3$

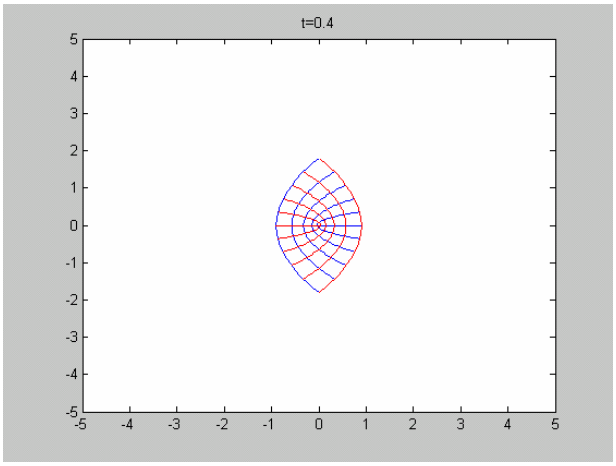

Figura 2.18 – $f(z) = tz^2$ para $t = 0.4$

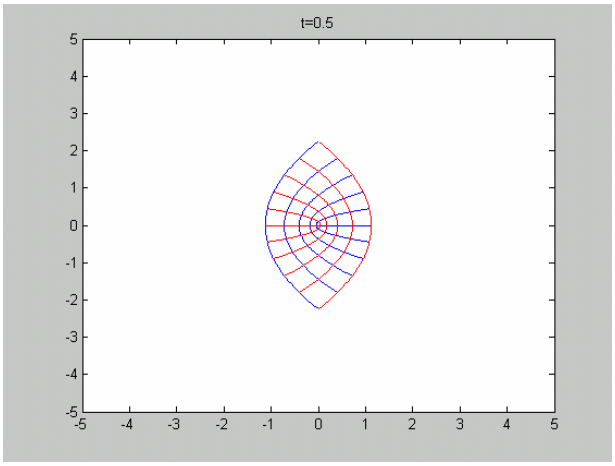

Figura 2.19 – $f(z) = tz^2$ para $t = 0.5$

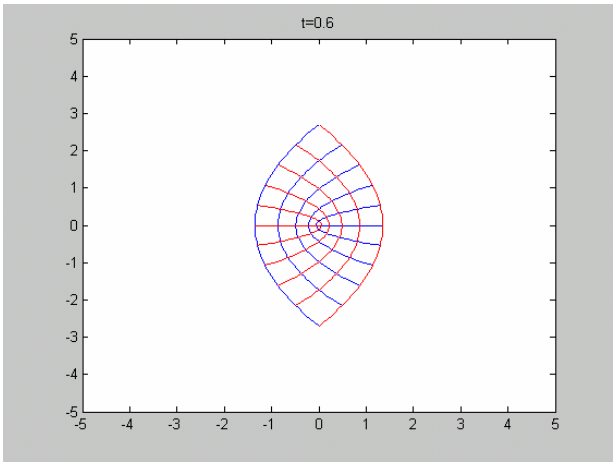

Figura 2.20 – $f(z) = tz^2$ para $t = 0.6$

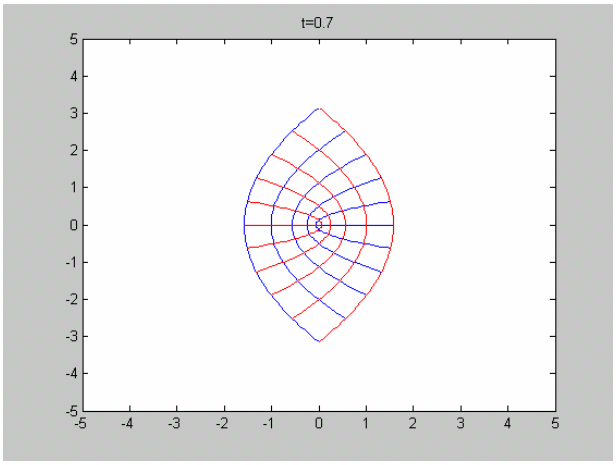

Figura2.21 – $f(z) = tz^2$ para $t = 0.7$

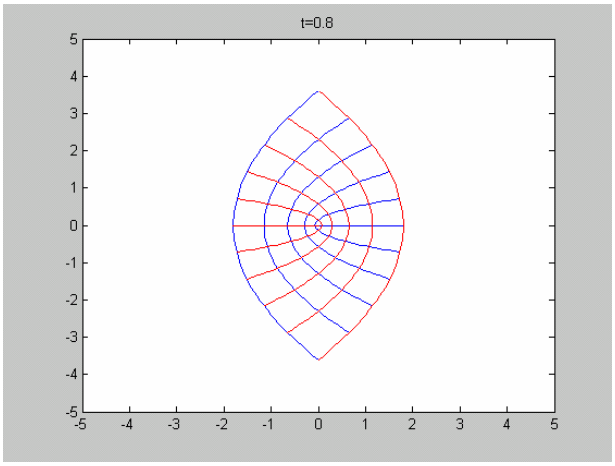

Figura 2.22 – $f(z) = tz^2$ para $t = 0.8$

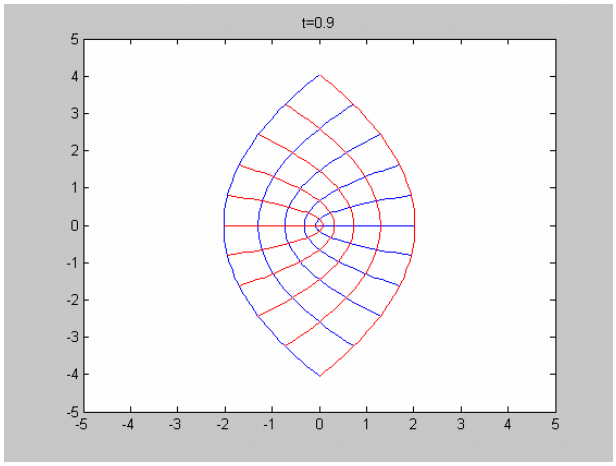

Figura 2.23 – $f(z) = tz^2$ para $t = 0.9$

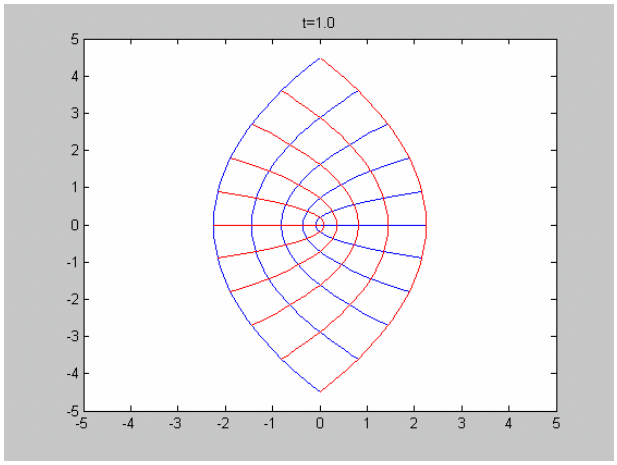

Figura 2.24 – $f(z) = tz^2$ para $t = 1.0$

Podemos notar que a medida que t aumenta as parábolas vão crescendo.

EXEMPLO 4: Se fizermos uma combinação do exemplo 2 e do exemplo 3, uma função do tipo $g(z) = (1-t)z + tz^2$, (vide Apêndice A, Programas 30 a 40) e variarmos t de 0 a 1, com um incremento de 0.1, teremos (Figuras 2.25 a 2.35):

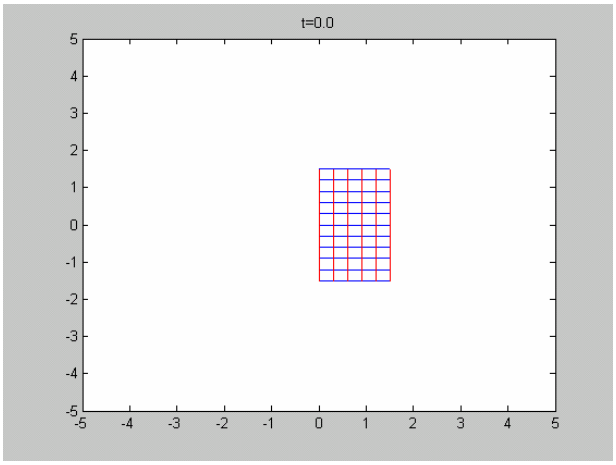

Figura 2.25 – $g(z) = (1-t)z + tz^2$ para t = 0.0

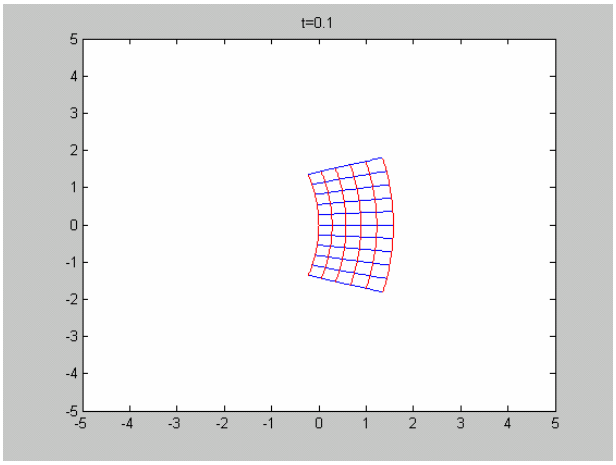

Figura 2.26 – $g(z) = (1-t)z + tz^2$ para t = 0.1

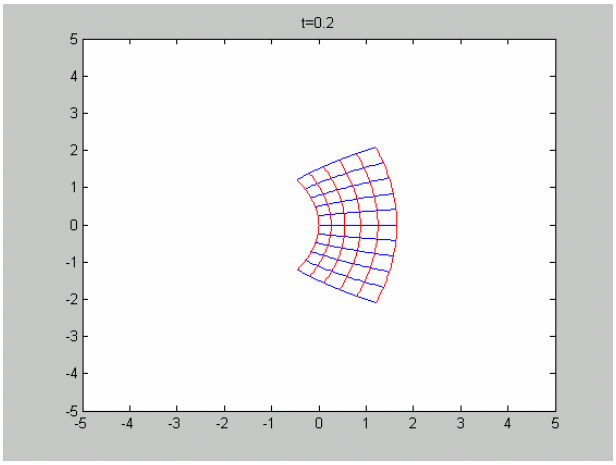

Figura 2.27 – $g(z) = (1-t)z + tz^2$ para t = 0.2

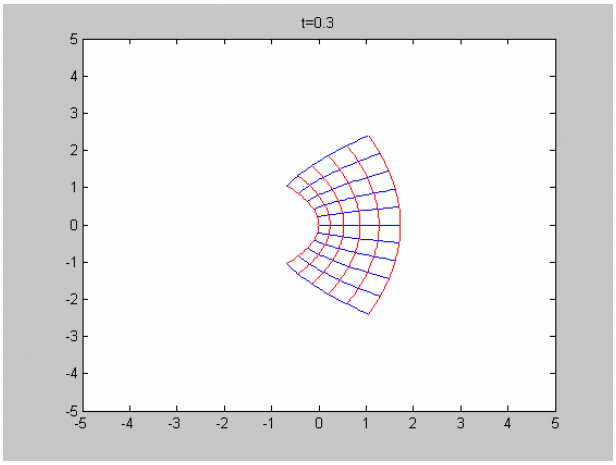

Figura 2.28 – $g(z) = (1-t)z + tz^2$ para t = 0.3

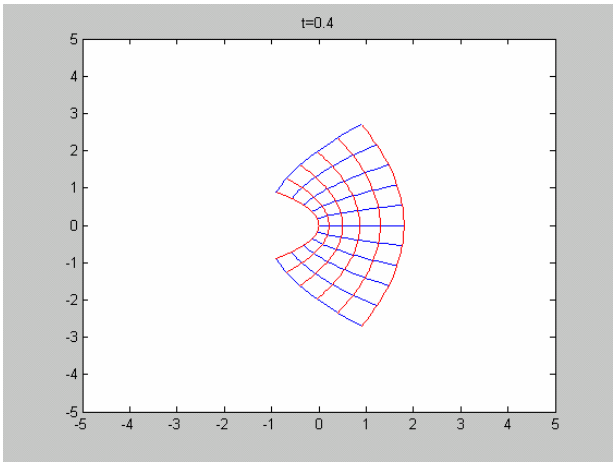

Figura 2.29 – $g(z) = (1-t)z + tz^2$ para t = 0.4

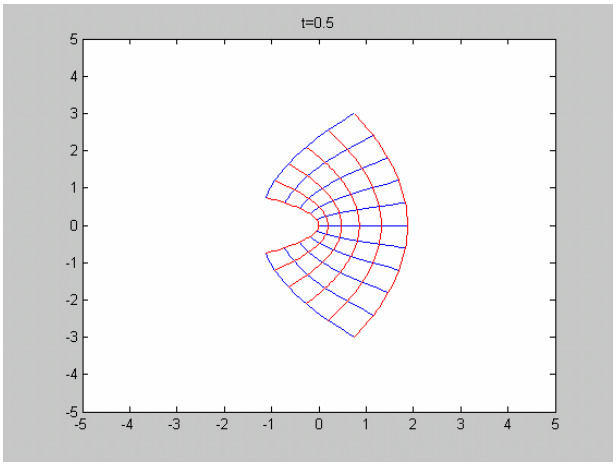

Figura 2.30 – $g(z) = (1-t)z + tz^2$ para t = 0.5

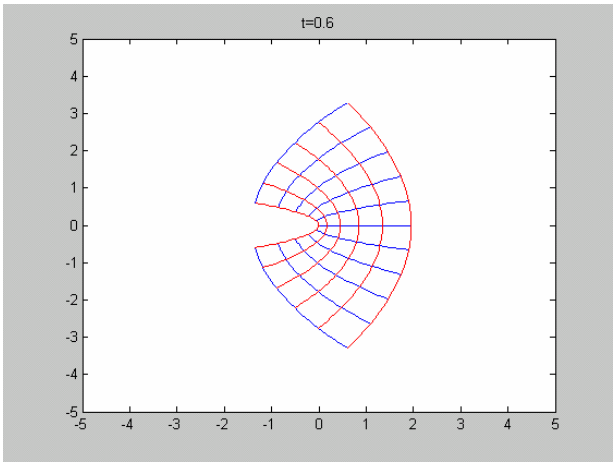

Figura 2.31 – $g(z) = (1-t)z + tz^2$ para t = 0.6

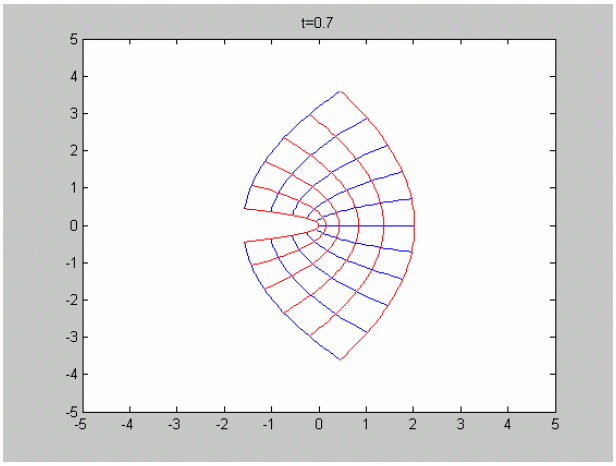

Figura 2.32 – $g(z) = (1-t)z + tz^2$ para t = 0.7

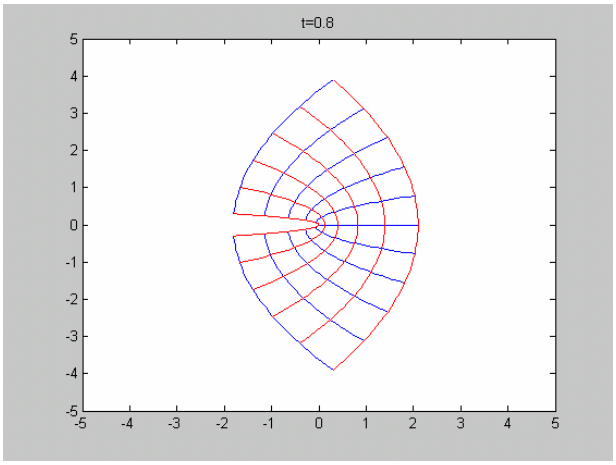

Figura 2.33 – $g(z) = (1-t)z + tz^2$ para t = 0.8

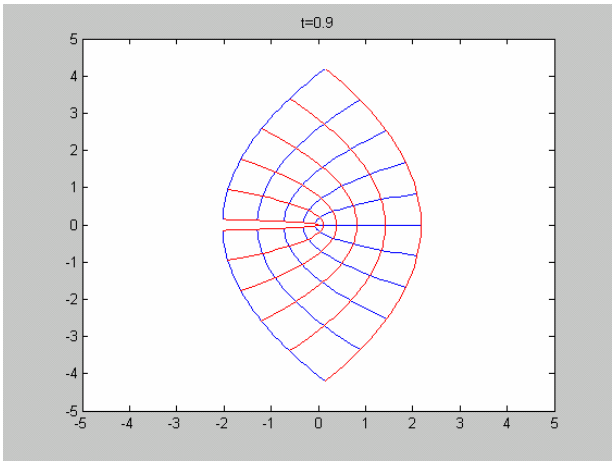

Figura 2.34 – $g(z) = (1-t)z + tz^2$ para t = 0.9

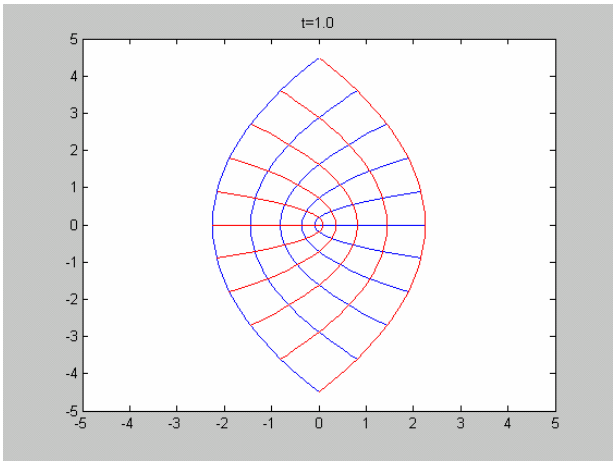

Figura 2.35 – $g(z) = (1-t)z + tz^2$ para t = 1.0

ANÁLISE DOS RESULTADOS: Vê-se dos vários exemplos acima que os mapeamentos complexos apresentam muitas propriedades interessantes e passíveis de serem usadas no 2º Modelo Analítico para aproximar os vários efeitos observados no 1º Método Numérico que transformam sucessivamente um círculo numa elipse, bananóide, amebóide, etc., a saber: translação, rotação, crescimento, flexão, torcimento, etc. Estamos estudando funções ou combinação de funções que descreverão melhor os vários efeitos observados no 1º Método Numérico.

CAPÍTULO 3 – ESTUDO DA CURVATURA DOS EIXOS MAIORES DA ELIPSE PARA O 2º MODELO ANALÍTICO (GEOMÉTRICO)

O estudo da curvatura dos eixos maiores de uma elipse apresenta resultados interessantes e passíveis de serem usadas no 2º Modelo Analítico para aproximar os vários efeitos observados no 1º Método Numérico.

Primeiramente iremos considerar uma elipse de semi-eixo maior (a), semi-eixo menor (b). Seccionando a elipse em n secções, obteremos vários segmentos de reta (a_n), que são paralelos ao semi-eixo maior original (a).

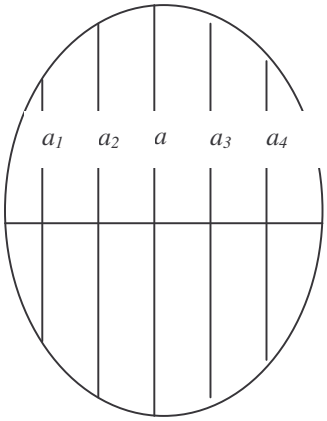

Figura 3.1 – Elipse seccionada

Onde a_1, a_2, a_3 e a_4 são os segmentos de reta (a_n), e a é o semi-eixo maior original.

Para curvar o semi-eixo maior (a) e os segmentos de reta (a_n), precisamos das coordenadas do centro de curvatura (x_0, y_0), do módulo do raio de curvatura (r) e do número de secções (n).

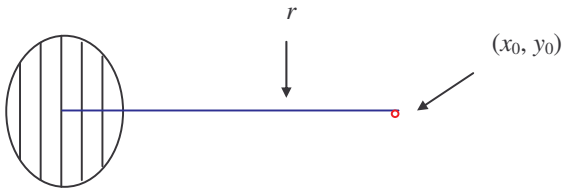

Figura 3.2 – Raio (r) e Centro de Curvatura (x_0, y_0)

Desejamos curvar o semi-eixo maior original (a), de modo que o semi-eixo maior curvado possua o mesmo comprimento do semi-eixo maior original (a).

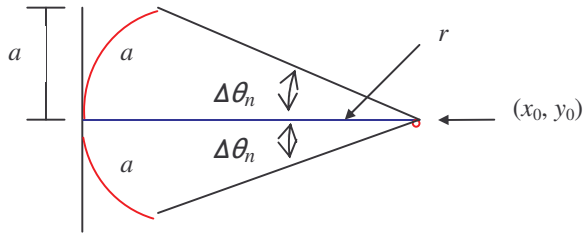

Figura 3.3 – Semi - Eixo Maior Curvado

Para isso ocorrer o ângulo de curvatura $\Delta\theta_{máx}$, deverá ser de:

$$\Delta\theta_{máx} = \frac{a}{r} \qquad (3.1)$$

Onde a é o semi-eixo maior original e r é o raio de curvatura.

Para os segmentos de reta (a_n) que são paralelos ao semi-eixo maior original (a), os novos ângulos de curvatura ($\Delta\theta_n$) devem ser proporcionais ao ângulo de curvatura para o semi- eixo maior original ($\Delta\theta_{máx}$), logo:

$$\frac{\Delta\theta_{máx}}{\Delta\theta_n} = \frac{a}{a_n} \qquad (3.2)$$

ou ainda:

$$\Delta\theta_n = \frac{a_n}{a}\Delta\theta_{máx} \qquad (3.3)$$

Após acharmos os ângulos correspondentes para cada segmento de reta, desenhamos o arco correspondente ao ângulo de curvatura utilizando o raio correspondente a cada segmento de reta, para obter então vários segmentos de reta curvados, que se aproximam da forma de um bananóide.

EXEMPLO 1: Se curvarmos uma elipse com semi – eixo maior $a = 3$, semi – eixo menor $b = 1$, as coordenadas do centro de atração são $(x_0,y_0) = (1,-5)$, o raio de curvatura $r = 5$ e o número de secções $n = 8$ (vide Apêndice B, Programa elipse_curva1) teremos (Figura 3.4):

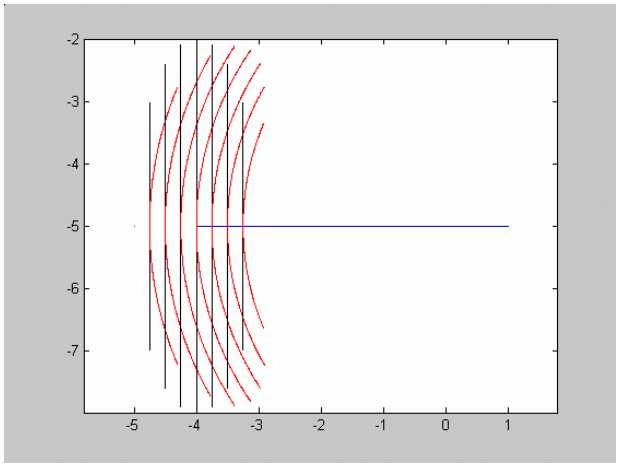

Figura 3.4 – Elipse Curvada 1

EXEMPLO 2: Se curvarmos uma elipse com semi – eixo maior $a = 5$, semi – eixo menor $b = 1.02$, as coordenadas do centro de atração são $(x_0,y_0) = (2,-3)$, o raio de curvatura $r = 3$ e o número de secções $n = 8$ (vide Apêndice B, Programa elipse_curva2) teremos (Figura 3.5):

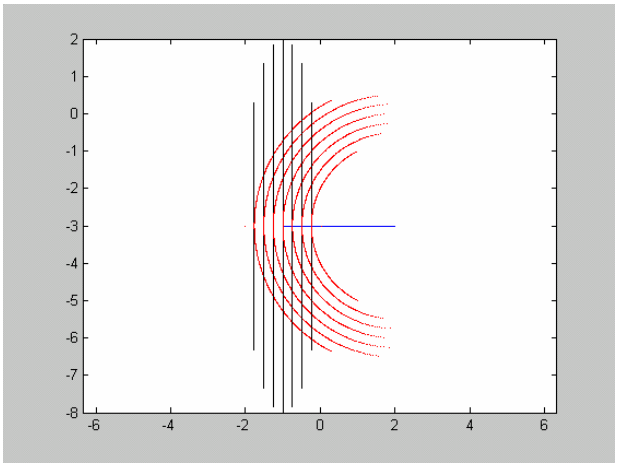

Figura 3.5 – Elipse Curvada 2

CAPÍTULO 4 – APROXIMAÇÃO DO BANANÓIDE PELA ELIPSE CURVADA COM O MÉTODO ITERATIVO

Ajustando uma elipse curvada ao bananóide obtido pelo Método Numérico no instante t = 5s (discutido no nosso Relatório de Iniciação Científica no período de fevereiro a julho de 2006), podemos notar que o bananóide no instante t = 5s pode ser aproximado manualmente pela elipse curvada.

EXEMPLO 1: Se sobrepormos uma elipse curvada com semi – eixo maior a = 416, semi – eixo menor b = 124.8, as coordenadas do centro de atração são (x_0,y_0) = (55,10), o raio de curvatura r = 312 e o número de secções n = 8 (vide Apêndice B, Programa bananoide_sobreposto1) teremos (Figura 4.1):

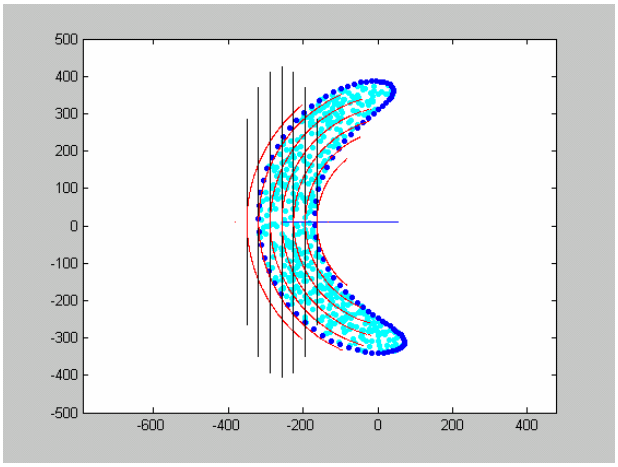

Figura 4.1 – Sobreposição da Elipse Curvada ao Bananóide 1

EXEMPLO 2: Se sobrepormos uma elipse curvada com semi – eixo maior a = 520, semi – eixo menor b = 104, as coordenadas do centro de atração são (x_0,y_0) = (55,10), o raio de curvatura r = 312 e o número de secções n = 8 (vide Apêndice B, Programa bananoide_sobreposto2) teremos (Figura 4.2):

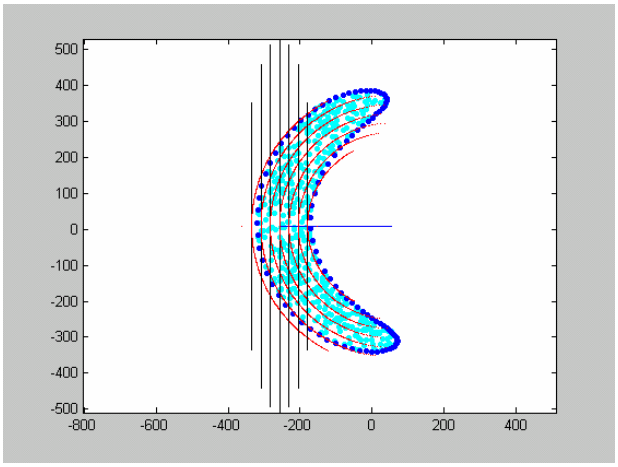

Figura 4.2 – Sobreposição da Elipse
Curvada ao Bananóide 2

CAPÍTULO 5 – AJUSTE DE UMA ELIPSE A UM CONJUNTO DE PONTOS PELO MÉTODO DOS MÍNIMOS QUADRADOS

No capítulo anterior, os valores do raio de curvatura, semi-eixo vertical e horizontal foram aproximados por tentativa. Nesta seção será feito o estudo de aproximações de funções a pontos iniciais para a escolha dos melhores parâmetros pelos métodos dos Mínimos Quadrados em Lotes utilizando como principal referência *Kuga, H.C.*

5.1 – MÉTODO DOS MINÍMOS QUADRADOS

O Método dos Mínimos Quadrados é um dos estimadores de parâmetros mais utilizados pela comunidade científica. Formalmente, o algoritmo trata de minimizar uma função custo (perda, índice de desempenho) do quadrado dos resíduos (vêm daí a razão do nome) na forma:

$$L = (y - Hx)^t (y - Hx) \qquad (5.1)$$

Ou na notação de norma vetorial:

$$L = \|y - Hx\|^2 \qquad (5.2)$$

Onde y representa o vetor contendo m medidas, x representa o vetor de n parâmetros a serem estimados, e H é uma matriz n x m que relaciona as medidas aos parâmetros. Ou seja:

$$y = Hx \qquad (5.3)$$

Ou ainda, de modo explícito:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} \qquad (5.4)$$

Onde $x = (a,b)$ é o vetor que contem os parâmetros a serem estimados.

5.2 – MÍNIMOS QUADRADOS EM LOTES

Em princípio, a forma mais direta de processar medidas é o chamado processamento em lotes ("batch"). Lembrando que *m* é o número de medidas:

• $m = 1$ → então solução indeterminada, lembrando que por um ponto passam infinitas retas.

• $m = 2$ → então ficamos com:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} t_1 & 1 \\ t_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \qquad (5.5)$$

cuja solução é imediata:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} t_1 & 1 \\ t_2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \qquad (5.6)$$

Obs: Se $t_1 \rightarrow t_2$ a matriz $\boldsymbol{H}$ tende a ser singular.

• $m > 2$ então ficamos com a solução clássica de mínimos quadrados, também chamada de equação normal:

$$\hat{x} = \left(H^t H\right)^{-1} H^t y \qquad (5.7)$$

Este último caso, que é o que nos interessa, resolve o problema através da formulação "batch", em lotes. Em outras palavras, este algoritmo processa as medidas todas de uma só vez.

A matriz $P = \left(H^t H\right)^{-1}$ dá uma idéia do erro nas estimativas, ela é composta de variâncias e correlações:

$$P = \left(H^t H\right)^{-1} = \begin{bmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ab} & \sigma_b^2 \end{bmatrix} \qquad (5.8)$$

onde σ^2 é o quadrado do desvio – padrão (variância), e σ_{ab} é o coeficiente de correlação entre *a* e *b*.

A covariância deve ser sempre definida positiva, ou seja, ser inversível, não-singular, para que possa existir a solução de mínimos quadrados.

Particularizando para nosso problema, temos:

Da equação da elipse temos:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1 \qquad (5.9)$$

logo:

$$y^2=b^2\left(1-\frac{x^2}{a^2}\right) \qquad (5.10)$$

ou ainda,

$$y^2=\left(b^2-\frac{b^2}{a^2}x^2\right) \qquad (5.11)$$

que também pode ser escrito como:

$$\left[y^2\right]=\left[x^2 \quad 1\right]\begin{bmatrix}-\frac{b^2}{a^2}\\ \\ b^2\end{bmatrix} \qquad (5.12)$$

ou ainda,

$$\begin{bmatrix}y_1^{\,2}\\ y_2^{\,2}\\ y_3^{\,2}\\ \vdots\\ y_i^{\,2}\end{bmatrix}=\begin{bmatrix}x_1^{\,2} & 1\\ x_2^{\,2} & 1\\ x_3^{\,2} & 1\\ \vdots & \vdots\\ x_i^{\,2} & 1\end{bmatrix}.\begin{bmatrix}-\frac{b^2}{a^2}\\ \\ b^2\end{bmatrix} \qquad (5.13)$$

$Y=X.A$, onde:

$$Y=\begin{bmatrix}y_1^{\,2}\\ y_2^{\,2}\\ y_3^{\,2}\\ \vdots\\ y_i^{\,2}\end{bmatrix}, \qquad X=\begin{bmatrix}x_1^{\,2} & 1\\ x_2^{\,2} & 1\\ x_3^{\,2} & 1\\ \vdots & \vdots\\ x_i^{\,2} & 1\end{bmatrix} \quad \text{e} \quad A=\begin{bmatrix}-\frac{b^2}{a^2}\\ \\ b^2\end{bmatrix} \qquad (5.14)$$

Utilizando os valores iniciais de x e y, x_i e y_i , obteremos os valores dos semi-eixos horizontal e vertical, respectivamente a e b, multiplicando a matriz pseudoinversa de X , $X^+(X^+ = (X^T.X)^{-1}.X^T)$, em ambos os lados da igualdade:

$$X^+.Y = X^+.X \qquad (5.15)$$

$$X^+.Y = (X^+.X).A \qquad (5.16)$$

Lembrando que $X^+.X$ é igual a matriz identidade, I, podemos dizer que:

$$X^+.Y = I.A \qquad (5.17)$$

$$X^+.Y = A \qquad (5.18)$$

Encontrando a Matriz A, podemos encontrar os semi-eixos horizontal e vertical, a e b respectivamente, com isso teremos a equação da elipse, centrada em (0,0), que mais se aproxima dos pontos iniciais.

Podemos ainda calcular o erro (ξ_i) que é dado por:

$$\xi_i = \frac{x_i^2}{a^2} + \frac{y_i^2}{b^2} - 1 \qquad (5.19)$$

EXEMPLO 1: Se escolhermos pontos inicias que pertencem a uma elipse (vide Apêndice B, Programa pseudo_inv1) teremos (Figura 5.1):

Tabela 5.1 – Pontos inciais que pertencem a uma elipse

i	x_i	y_i	ξ_i
1	-3	(5/4).√7	0
2	-1	-(5/4).√15	0
3	1	(5/4).√15	0
4	2	-(5/2).√3	0

Utilizando o MATLAB:

```
>> pseudo_inv1

b =

    5


a =
```

```
    4


Erro =

  1.0e-015 *

   -0.1110
         0
         0
         0

>>
```

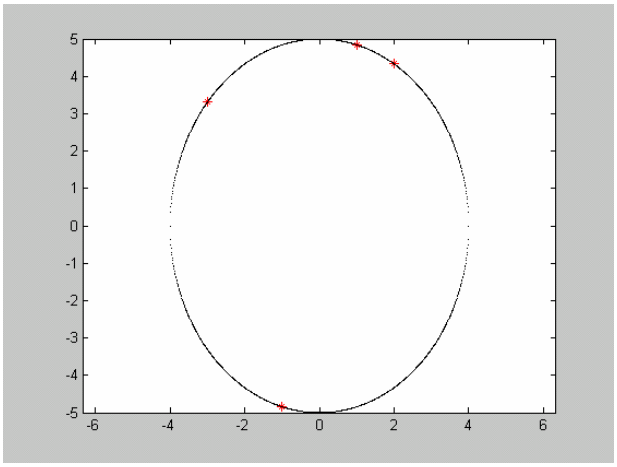

Figura 5.1 – Elipse ajustada 1

EXEMPLO 2: Se utilizarmos pontos inicias que estão próximos a uma elipse (vide Apêndice B, Programa pseudo_inv2) teremos (Figura 5.2):

Tabela 5.2 – Pontos inciais que estão próximos a uma elipse

i	x_i	y_i	ξ_i
1	-3	3,35	0.0742
2	-1	-5	-0.1375
3	1	4,5	0.0912
4	2	4,6	-0.0278

Utilizando o MATLAB:

```
>> pseudo_inv2

b =

    5.0187


a =

    4.1340


Erro =

    0.0742
   -0.1375
    0.0912
   -0.0278


>>
```

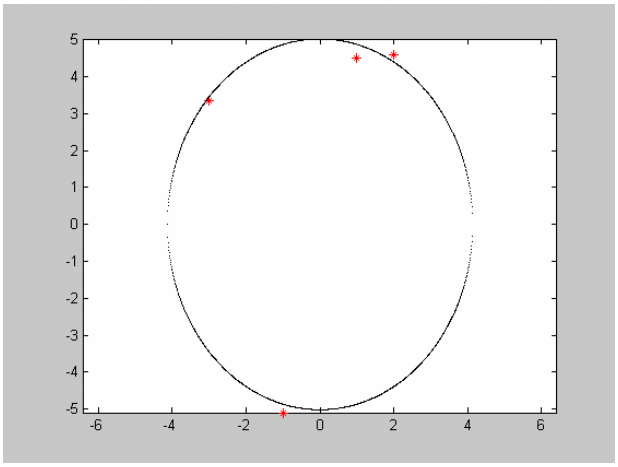

Figura 5.2 – Elipse ajustada 2

EXEMPLO 3: Se utilizarmos pontos inicias que estão distantes a uma elipse (vide Apêndice B, Programa pseudo_inv3) teremos (Figura 5.3):

Tabela 5.3 – Pontos inciais que estão distantes a uma elipse

i	x_i	y_i	ξ_i
1	-3	4,65	0.2388
2	-1	-5,9	-0.4319
3	1	3,9	0.2827
4	2	5,7	-0.0895

Utlizando o MATLAB:

```
>> pseudo_inv3

b =

    5.2373


a =

    8.5838


Erro =

    0.2388
   -0.4319
    0.2827
   -0.0895

>>
```

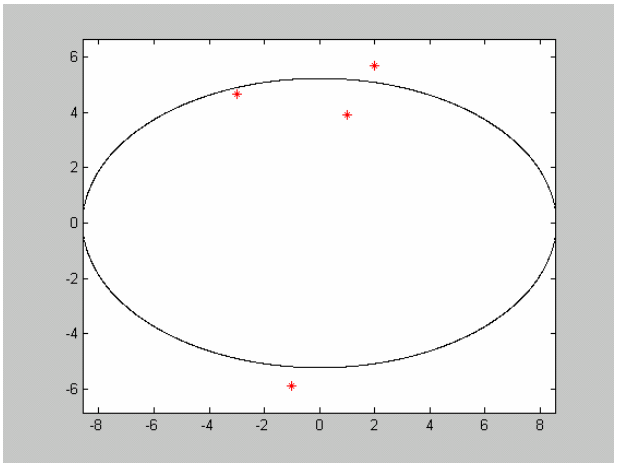

Figura 5.3 – Elipse ajustada 3

ANÁLISE DOS RESULTADOS: Vê-se dos vários exemplos acima que o ajuste de uma elipse a um conjunto de pontos pelo Método dos Mínimos Quadrados apresentam propriedades passíveis de serem usadas no 2º Modelo Analítico para aproximar os vários efeitos observados no 1º Método Numérico que transformam sucessivamente um círculo numa elipse, bananóide, amebóide, etc., a saber: translação, rotação, crescimento, flexão, torcimento, etc.

CAPÍTULO 6 – CONCLUSÕES, COMENTÁRIOS E SUGESTÕES PARA O PROSSEGUIMENTO DO TRABALHO

O trabalho foi idealizado a partir das observações dos resultados dos Projetos de Pesquisa precedentes, nos quais notou-se que a propagação de Detritos Espaciais ocorria segundo a forma de uma elipse progressivamente deformada ("bananóide"), cujos eixos cresciam segundo alguma taxa, ao mesmo tempo em que a elipse girava em torno do seu Centro de Massa – CM , e este girava em torno de um ponto (provavelmente o Centro de Atração da Terra) segundo a órbita inicial.

Os programas foram compreendidos, executados e testados com êxito e foram dados vários passos para a construção do 2^{o} modelo analítico da propagação dos detritos espaciais, pois foi com o curvamento dos eixos maiores de uma elipse que obtivemos as figuras com a forma mais aproximada, até agora, de um bananóide, para podermos então compará-las com as figuras obtidas com o 1º Modelo Numérico, obtido pelas 3 Leis de Kepler e pela Equação de Kepler para cada detrito após a fragmentação.

Futuramente serão ajustados os parâmetros do 2º modelo analítico ao 1º modelo numérico (por iterações, e depois pelo método dos mínimos quadrados, etc.) para que este simule da melhor maneira possível a propagação de detritos espaciais.

Poderemos assim analisar os problemas de colisão e interferência dos detritos espaciais com outros objetos encontrados no espaço como: satélites, ônibus espaciais e estações espaciais.

REFERÊNCIAS BIBLIOGRÁFICAS

1) KUGA, H.K., RAO, K.R., *Introdução à Mecânica Orbital,* INPE, São José dos Campos – SP, 1995.

2) JOHNSON, N.L., & MCKINIGHT, D.S. *Artificial Space Debris (Updated Edition).* Krieger Pub. Co., Malabar, FL, USA, 1991.

3) CHOBOTOV, V.A. (ed.) *Orbital Mechanics (2 Ed.)* Reston, VA, USA, AIAA, 1996.

4) SOUZA, M.L.O., NUNES, D., *Forecasting Space Debris Distribution: A Measure Theory Approach,* 51st. International Astronautical Congress – IAC. Rio de Janeiro - RJ, 2-6 Out.2000, Paper IAA-00-IAA.6.4.07.

5) ROSSER, J.B. (ED.) *Space Mathematics, Part I.* American Mathematical Society, New York, NY, USA, 1996.

6) CHANDRASEKHAR, S. *Principles of Stellar Dynamics*. Chicago Univ. Press, Chicago, IL, USA, 1942; e Dover Pub, New York, NY, USA, 1960.

7) CARNAHAN, B.; LUTHER, H. A.; WILKES, J. O. *Applied Numerical Methods/*. New York : John Wiley, c1969 604 p. : ISBN 471.135070 (enc.)

8) PRODENGE. *Curso de MATLAB 5.1; Introdução à Solução de Problemas de Engenharia (Apostila, 2a edição)*; Programa Prodenge/ Sub-Programa Reenge; Universidade do Estado do Rio de Janeiro.

9) HANSELMAN, D; LITTLEFIELD, B. *Versão do Estudante MATLAB 5 – Guia do Usuário*. São Paulo, MAKRON Books, 1999.

10) KUGA, H. K. *Otimização em Sistemas Dinâmicos II,* INPE, São José dos Campos.

11) http://www.ima.umn.edu/~arnold/complex.html, acessado em 07/11/2006.

12) http://www.arauto.uminho.pt/pessoas/EPereira/Portugues/Cadeiras/FisicaI-LESI/Problemas/ ComputacaoParticula-FisI.pdf, acessado em 16/01/2007

APÊNDICE A: PROGRAMAS PARA MAPEAMENTOS COMPLEXOS DESENVOLVIDOS POR ANDREZA DA COSTA BATISTA UTILIZANDO O MATLAB

Programa 6

```
% metade do quadrado localizado no semi-plano direito
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=(y*i+x);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=(y1*i+x1);
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
```

Programa 7

```
%mapeamento: f(z) = z^2
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=(y*i+x).^2;
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=(y1*i+x1).^2;
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
```

Programa 8

```
% "(1-t)z" mapeamento para t=0.0
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=1.0*(y*i+x);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=1.0*(y1*i+x1);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.0');
```

Programa 9

```
% "(1-t)z" mapeamento para t=0.1
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.9*(y*i+x);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.9*(y1*i+x1);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.1');
```

Programa 10

```
% "(1-t)z" mapeamento para t=0.2
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.8*(y*i+x);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.8*(y1*i+x1);
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.2');
```

Programa 11

```
% "(1-t)z" mapeamento para t=0.3
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.7*(y*i+x);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.7*(y1*i+x1);
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.3');
```

Programa 12

```
% "(1-t)z" mapeamento para t=0.4
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.6*(y*i+x);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.6*(y1*i+x1);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.4');
```

Programa 13

```
% "(1-t)z" mapeamento para t=0.5
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.5*(y*i+x);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.5*(y1*i+x1);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.5');
```

Programa 14

```
% "(1-t)z" mapeamento para t=0.6
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.4*(y*i+x);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.4*(y1*i+x1);
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.6');
```

Programa 15

```
% "(1-t)z" mapeamento para t=0.7
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.3*(y*i+x);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.3*(y1*i+x1);
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.7');
```

Programa 16

```
% "(1-t)z" mapeamento para t=0.8
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.2*(y*i+x);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.2*(y1*i+x1);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.8');
```

Programa 17

```
% "(1-t)z" mapeamento para t=0.9
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.1*(y*i+x);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.1*(y1*i+x1);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.9');
```

Programa 18

```
% "(1-t)z" mapeamento para t=1.0
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.0*(y*i+x);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.0*(y1*i+x1);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=1.0');
```

Programa 19

```
% "t*z*z" mapeamento para t=0.0
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.0*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.0*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.0');
```

Programa 20

```
% "t*z*z" mapeamento para t=0.1
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.1*((y*i+x).^2);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.1*(y1*i+x1).^2;
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.1');
```

Programa 21

```
% "t*z*z" mapeamento para t=0.2
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.2*((y*i+x).^2);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.2*(y1*i+x1).^2;
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.2');
```

Programa 22

```
% "t*z*z" mapeamento para t=0.3
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.3*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.3*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.3');
```

Programa 23

```
% "t*z*z" mapeamento para t=0.4
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.4*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.4*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.4');
```

Programa 24

```
% "t*z*z" mapeamento para t=0.5
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.5*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.5*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.5');
```

Programa 25

```
% "t*z*z" mapeamento para t=0.6
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.6*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.6*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.6');
```

Programa 26

```
% "t*z*z" mapeamento para t=0.7
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.7*((y*i+x).^2);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.7*(y1*i+x1).^2;
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.7');
```

Programa 27

```
% "t*z*z" mapeamento para t=0.8
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.8*((y*i+x).^2);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.8*(y1*i+x1).^2;
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.8');
```

Programa 28

```
% "t*z*z" mapeamento para t=0.9
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.9*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.9*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.9');
```

Programa 29

```
% "t*z*z" mapeamento para t=1.0
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=1.0*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=1.0*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=1.0');
```

Programa 30

```
% mapeamento para t=0.0
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=1*(y*i+x)+0*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=1*(y1*i+x1)+0*((y1*i+x1).^2);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.0')
```

Programa 31

```
% mapeamento para t=0.1
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.9*(y*i+x)+0.1*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.9*(y1*i+x1)+0.1*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.1');
```

Programa 32

```
% mapeamento para t=0.2
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.8*(y*i+x)+0.2*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.8*(y1*i+x1)+0.2*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.2');
```

Programa 33

```
% mapeamento para t=0.3
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.7*(y*i+x)+0.3*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.7*(y1*i+x1)+0.3*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.3');
```

Programa 34

```
% mapeamento para t=0.4
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.6*(y*i+x)+0.4*((y*i+x).^2);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.6*(y1*i+x1)+0.4*(y1*i+x1).^2;
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.4');
```

Programa 35

```
% mapeamento para t=0.5
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.5*(y*i+x)+0.5*((y*i+x).^2);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.5*(y1*i+x1)+0.5*(y1*i+x1).^2;
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.5')
```

Programa 36

```
% mapeamento para t=0.6
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.4*(y*i+x)+0.6*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.4*(y1*i+x1)+0.6*(y1*i+x1).^2;
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.6')
```

Programa 37

```
% mapeamento para t=0.7
for y=-1.5:0.3:1.5;
     x=0:0.05:1.5;
     w=0.3*(y*i+x)+0.7*((y*i+x).^2);
     plot(real(w),imag(w),'b')
     hold on

end

for x1=0:0.3:1.5;
     y1=-1.5:0.05:1.5;
     w1=0.3*(y1*i+x1)+0.7*((y1*i+x1).^2);
     plot(real(w1),imag(w1),'r')
     hold on
end
axis([-5 5 -5 5]);
title('t=0.7')
```

Programa 38

```
% mapeamento para t=0.8
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.2*(y*i+x)+0.8*((y*i+x).^2);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.2*(y1*i+x1)+0.8*((y1*i+x1).^2);
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.8')
```

Programa 39

```
% mapeamento para t=0.9
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0.1*(y*i+x)+0.9*((y*i+x).^2);
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0.1*(y1*i+x1)+0.9*((y1*i+x1).^2);
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=0.9')
```

Programa 40

```
% mapeamento para t=1.0
for y=-1.5:0.3:1.5;
    x=0:0.05:1.5;
    w=0*(y*i+x)+1*(y*i+x).^2;
    plot(real(w),imag(w),'b')
    hold on

end

for x1=0:0.3:1.5;
    y1=-1.5:0.05:1.5;
    w1=0*(y1*i+x1)+1*(y1*i+x1).^2;
    plot(real(w1),imag(w1),'r')
    hold on
end
axis([-5 5 -5 5]);
title('t=1.0')
```

APÊNDICE B: PROGRAMAS PARA A CURVATURA DOS EIXOS DE UMA ELIPSE , APROXIMAÇÃO DO BANANÓIDE PELA ELIPSE CURVADA COM O MÉTODO ITERATIVO E PARA O AJUSTE DE UMA ELIPSE A UM CONJUNTO DE PONTOS PELO MÉTODO DOS MÍNIMOS QUADRADOS DESENVOLVIDOS POR ANDREZA DA COSTA BATISTA UTILIZANDO O MATLAB

Programa elipse_curva1

```
%%%%%%% DADOS DE ENTRADA
xc=1;              %(xc,yc) coordenadas do centro de curvatura
yc=-5;
a=3;              %semi-eixo maior
b=1;              %semi-eixo menor
raio=5;           %raio de curvatura
teta=(pi);        %inclinacao do raio em relacao ao eixo x
nf=8;             %numero (par) de fatias da elipse


%centro da elipse (xce,yce)
xce=xc+raio*cos(teta);
yce=yc+raio*sin(teta);

%plotagem do raio para teta=pi
for xraio=xce:0.01:xc;
   yraio=yce;
   plot(xraio,yraio)
   hold on
end
axis equal


%%%%%%%plotagem das fatias da elipse


for xcee=(xce-b):(b/(nf/2)):(xce+b);
   ye1=yce-a*sqrt(1-((xcee-xce).^2)/(b.^2));
   ye2=yce+a*sqrt(1-((xcee-xce).^2)/(b.^2));

      for yee=ye1:0.01:ye2;
         xee=xcee;
         plot(xee,yee,'-k')
         hold on
```

```
        end

         %%%%%%%% entortamento


        an=(ye2-ye1)/2;             %novos semi-eixos maiores
        man=abs(an);                %modulo dos novos semi-eixos
        detetamax=a/raio;           %delta teta maximo
        detetan=(detetamax*an)/a;   %delta teta proporcional aos novos semi-eixos
maiores
        raion=xcee-xc;              %novos raios

        yctorto=yc+raion*sin(pi-detetan);      % coordenadas de x e y apos o
entortamento
        xctorto=xc-raion*cos(pi-detetan);

        yctorto1=yc+raion*sin(pi+detetan);
        xctorto1=xc-raion*cos(pi+detetan);


          %%%%%%%%%%%%plotagem da circunferencia
        for ycirc1=yctorto:0.01:yctorto1
            xcirc1=xc-sqrt(raion.^2-((ycirc1-yc).^2));
            plot(xcirc1,ycirc1,'-r')
            hold on
        end

end

axis equal
```

Programa elipse_curva2

```
%%%%%%% DADOS DE ENTRADA
xc=2;              %(xc,yc) coordenadas do centro de curvatura
yc=-3;
a=5;              %semi-eixo maior
b=1.02;               %semi-eixo menor
raio=3;            %raio de curvatura
teta=(pi);         %inclinacao do raio em relacao ao eixo x
nf=8;              %numero (par) de fatias da elipse


%centro da elipse (xce,yce)
xce=xc+raio*cos(teta);
yce=yc+raio*sin(teta);

%plotagem do raio para teta=pi
for xraio=xce:0.01:xc;
   yraio=yce;
   plot(xraio,yraio)
   hold on
end
axis equal


%%%%%%%plotagem das fatias da elipse


for xcee=(xce-b):(b/(nf/2)):(xce+b);
   ye1=yce-a*sqrt(1-((xcee-xce).^2)/(b.^2));
   ye2=yce+a*sqrt(1-((xcee-xce).^2)/(b.^2));

      for yee=ye1:0.01:ye2;
         xee=xcee;
         plot(xee,yee,'-k')
         hold on
      end

       %%%%%%%%% entortamento


      an=(ye2-ye1)/2;            %novos semi-eixos maiores
      man=abs(an);               %modulo dos novos semi-eixos
```

```
        detetamax=a/raio;           %delta teta maximo
        detetan=(detetamax*an)/a;   %delta teta proporcional aos novos semi-eixos
maiores
        raion=xcee-xc;              %novos raios

        yctorto=yc+raion*sin(pi-detetan);      % coordenadas de x e y apos o
entortamento
        xctorto=xc-raion*cos(pi-detetan);

        yctorto1=yc+raion*sin(pi+detetan);
        xctorto1=xc-raion*cos(pi+detetan);


         %%%%%%%%%%%%plotagem da circunferencia
        for ycirc1=yctorto:0.01:yctorto1
           xcirc1=xc-sqrt(raion.^2-((ycirc1-yc).^2));
           plot(xcirc1,ycirc1,'-r')
           hold on
        end

end

axis equal
```

Programa bananoide_sobreposto1

```
load pos5.txt

plot(pos5(1:501,1),pos5(1:501,2),'c.')
hold on

plot(pos5(502:601,1),pos5(502:601,2),'b.')

%602 a 701 repetiçao dos ultimos 100 pontos em ordem inversa, ou seja, 601=602 e
502=701

axis ([-500 500 -500 500])

%%%%%%% DADOS DE ENTRADA
xc=55;              %(xc,yc) coordenadas do centro de curvatura
yc=10;
a=416;              %semi-eixo maior
b=124.8;            %semi-eixo menor
raio=312;           %raio de curvatura
teta=(pi);          %inclinacao do raio em relacao ao eixo x
nf=8;               %numero (par) de fatias da elipse


%centro da elipse (xce,yce)
xce=xc+raio*cos(teta);
yce=yc+raio*sin(teta);

%plotagem do raio para teta=pi
for xraio=xce:1:xc;
   yraio=yce;
   plot(xraio,yraio)
   hold on
end
axis equal


%%%%%%%plotagem das fatias da elipse


for xcee=(xce-b):(b/(nf/2)):(xce+b);
   ye1=yce-a*sqrt(1-((xcee-xce).^2)/(b.^2));
   ye2=yce+a*sqrt(1-((xcee-xce).^2)/(b.^2));

      for yee=ye1:1:ye2;
         xee=xcee;
```

```
        plot(xee,yee,'-k')
        hold on
    end

     %%%%%%%% entortamento


    an=(ye2-ye1)/2;             %novos semi-eixos maiores
    man=abs(an);                %modulo dos novos semi-eixos
    detetamax=a/raio;           %delta teta maximo
    detetan=(detetamax*an)/a;   %delta teta proporcional aos novos semi-eixos maiores
    raion=xcee-xc;              %novos raios

    yctorto=yc+raion*sin(pi-detetan);      % coordenadas de x e y apos o entortamento
    xctorto=xc-raion*cos(pi-detetan);

    yctorto1=yc+raion*sin(pi+detetan);
    xctorto1=xc-raion*cos(pi+detetan);


      %%%%%%%%%%%%plotagem da circunferencia
    for ycirc1=yctorto:1:yctorto1
        xcirc1=xc-sqrt(raion.^2-((ycirc1-yc).^2));
        plot(xcirc1,ycirc1,'-r')
        hold on
    end

end

axis equal
```

Programa bananoide_sobreposto2

```
load pos5.txt

plot(pos5(1:501,1),pos5(1:501,2),'c.')
hold on

plot(pos5(502:601,1),pos5(502:601,2),'b.')

%602 a 701 repetiçao dos ultimos 100 pontos em ordem inversa, ou seja, 601=602 e
502=701

%%%%%%% DADOS DE ENTRADA
xc=55;              %(xc,yc) coordenadas do centro de curvatura
yc=10;
a=520;              %semi-eixo maior
b=104;              %semi-eixo menor
raio=312;           %raio de curvatura
teta=(pi);          %inclinacao do raio em relacao ao eixo x
nf=8;               %numero (par) de fatias da elipse


%centro da elipse (xce,yce)
xce=xc+raio*cos(teta);
yce=yc+raio*sin(teta);

%plotagem do raio para teta=pi
for xraio=xce:1:xc;
   yraio=yce;
   plot(xraio,yraio)
   hold on
end
axis equal


%%%%%%%plotagem das fatias da elipse


for xcee=(xce-b):(b/(nf/2)):(xce+b);
   ye1=yce-a*sqrt(1-((xcee-xce).^2)/(b.^2));
   ye2=yce+a*sqrt(1-((xcee-xce).^2)/(b.^2));

      for yee=ye1:1:ye2;
```

```
            xee=xcee;
            plot(xee,yee,'-k')
            hold on
        end

         %%%%%%%% entortamento


        an=(ye2-ye1)/2;             %novos semi-eixos maiores
        man=abs(an);                %modulo dos novos semi-eixos
        detetamax=a/raio;           %delta teta maximo
        detetan=(detetamax*an)/a;   %delta teta proporcional aos novos semi-eixos
maiores
        raion=xcee-xc;              %novos raios

        yctorto=yc+raion*sin(pi-detetan);      % coordenadas de x e y apos o
entortamento
        xctorto=xc-raion*cos(pi-detetan);

        yctorto1=yc+raion*sin(pi+detetan);
        xctorto1=xc-raion*cos(pi+detetan);


          %%%%%%%%%%%%plotagem da circunferencia
        for ycirc1=yctorto:1:yctorto1
            xcirc1=xc-sqrt(raion.^2-((ycirc1-yc).^2));
            plot(xcirc1,ycirc1,'-r')
            hold on
        end

end

axis equal
```

Programa pseudo_inv1

```
x=[2;1;-1;-3];
y=[2.5*sqrt(3);(5/4)*sqrt(15);-(5/4)*sqrt(15);(5/4)*sqrt(7)];

X=[(x(1,1)).^2 1;(x(2,1)).^2 1;(x(3,1)).^2 1;(x(4,1)).^2 1];
Y=[(y(1,1)).^2;(y(2,1)).^2;(y(3,1))^2;(y(4,1)).^2];
A=pinv(X)*Y;
b=sqrt(A(2,1)) %semi-eixo vertical
a=sqrt(-A(2,1)/A(1,1)) %semi-eixo horizontal

figure
plot(x(1,1),y(1,1),'*r',x(2,1),y(2,1),'*r',x(3,1),y(3,1),'*r',x(4,1),y(4,1),'*r')
hold on

for xe=-a:0.01:a;
   ye1=b*sqrt(1-(xe.^2)/(a.^2));
   ye2=-b*sqrt(1-(xe.^2)/(a.^2));
   plot(xe,ye1,'k')
   hold on
   plot(xe,ye2,'k')
   hold on
end

axis equal
```

Programa pseudo_inv2

```
x=[2;1;-1;-3];
y=[4.6;4.5;-5.1;3.35];

X=[(x(1,1)).^2 1;(x(2,1)).^2 1;(x(3,1)).^2 1;(x(4,1)).^2 1];
Y=[(y(1,1)).^2;(y(2,1)).^2;(y(3,1))^2;(y(4,1)).^2];
A=pinv(X)*Y;
b=sqrt(A(2,1)) %semi-eixo vertical
a=sqrt(-A(2,1)/A(1,1)) %semi-eixo horizontal

figure
plot(x(1,1),y(1,1),'*r',x(2,1),y(2,1),'*r',x(3,1),y(3,1),'*r',x(4,1),y(4,1),'*r')
hold on

for xe=-a:0.01:a;
   ye1=b*sqrt(1-(xe.^2)/(a.^2));
   ye2=-b*sqrt(1-(xe.^2)/(a.^2));
   plot(xe,ye1,'k')
   hold on
   plot(xe,ye2,'k')
   hold on
end

axis equal
```

Programa pseudo_inv3

```
x=[2;1;-1;-3];
y=[5.7;3.9;-5.9;4.65];

X=[(x(1,1)).^2 1;(x(2,1)).^2 1;(x(3,1)).^2 1;(x(4,1)).^2 1];
Y=[(y(1,1)).^2;(y(2,1)).^2;(y(3,1))^2;(y(4,1)).^2];
A=pinv(X)*Y;
b=sqrt(A(2,1)) %semi-eixo vertical
a=sqrt(-A(2,1)/A(1,1)) %semi-eixo horizontal

figure
plot(x(1,1),y(1,1),'*r',x(2,1),y(2,1),'*r',x(3,1),y(3,1),'*r',x(4,1),y(4,1),'*r')
hold on

for xe=-a:0.01:a;
   ye1=b*sqrt(1-(xe.^2)/(a.^2));
   ye2=-b*sqrt(1-(xe.^2)/(a.^2));
   plot(xe,ye1,'k')
   hold on
   plot(xe,ye2,'k')
   hold on
end

axis equal
```

APÊNDICE C: MÉTODOS DE INTEGRAÇÃO NÚMERICA

C.1 – INTRODUÇÃO

Para haver um melhor entendimento dos métodos de integração numérica de soluções de equações diferenciais ordinárias, foi feito um estudo destes métodos utilizando como principal referência *Carnahan, Brice; Luther, H. A.; Wilkes, J. O.*, onde destacamos o 1º Método de Aproximação de Euler e o Método de Runge – Kutta de terceira e quarta ordem.

C.2 – 1º MÉTODO DE APROXIMAÇÃO DE EULER

O 1º Método de Euler é o método numérico mais elementar de resolução de equações diferenciais ordinárias. Tem uma interpretação geométrica elementar. Considere a equação diferencial:

$$\frac{dy}{dx} = f(x, y) \qquad \text{(C.1)}$$

onde: x e y são as variáveis independente e dependente, respectivamente; $f(x, y)$, é a função derivada que é em geral função das variáveis independente e dependente. A derivada pode ser definida como:

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} \qquad \text{(C.2)}$$

Sejam conhecidos os valores num determinado ponto (x_i, y_i), com $i \geq 0$. O valor de y_{i+1} pode ser determinado a partir da equação C.2 aproximando a derivada por diferenças finitas:

$$\lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \lim_{\Delta x \to 0} \frac{y_{i+1} - y_i}{h} \qquad \text{(C.3)}$$

onde $h = x_{i+1} - x_i$ é chamado passo (constante) de integração da equação diferencial. Se a derivada for escrita como $f(x_i, y_i)$, e se deixar cair o limite de um intervalo infinitesimal, fica:

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i) \qquad \text{(C.4)}$$

e finalmente:

$$y_{i+1} = y_i + hf(x_i, y_i) \qquad \text{(C.5)}$$

que é a equação que constitui o **1º Método de Euler** de resolução numérica de equações diferenciais ordinárias.

C.2.1 – INTERPRETAÇÃO GRÁFICA DO 1º MÉTODO DE APROXIMAÇÃO DE EULER

Para solucionar uma equação diferencial ordinária no intervalo $[x_0, x_1]$ supomos que a linha tangente de $y(x)$ em x_0 continua (veja Figura C.1). Se aplicado repetidamente em diversos intervalos na sequência, o 1º Método de Euler traça fora da curva segmentos de retas com inclinações f_i, i=0, 1, 2, ..., n – 1. Na Figura C.1 está uma interpretação gráfica deste Método:

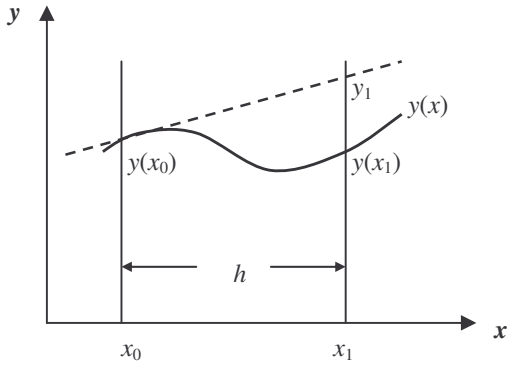

Figura C.1 – 1º Método de Euler

Ex: Consideremos a equação diferencial:

$$f(x, y) = \frac{dy}{dx} = x + y \qquad \text{(C.6)}$$

Dada a condição inicial $x_0 = 0$, $y(x_0) = y_0 = 0$, a solução analítica é:

$$y = e^x - x - 1 \qquad \text{(C.7)}$$

A solução desta mesma equação diferencial utilizando o 1º Método de Euler está exibida na Tabela C.1, onde foi usado um passo $h = 0.1$, e um limite superior de integração $x_{10} = 1.0$. A coluna 3 contém os valores de y_i que é calculado pelo 1º Método de aproximação de Euler:

$$y_{i+1} = y_i + hf(x_i, y_i) \quad \text{, com } i \geq 0 \qquad \text{(C.8)}$$

A coluna 4 contém os valores de $f(x_i, y_i)$ calculados de acordo com o y_i calculado na coluna anterior. A coluna 5 contém os valores de $y(x_i)$ calculados e arredondados de acordo com a solução analítica (C.7). A coluna 6 mostra o erro global (e) arredondado onde $e = y_i - y(x_i)$. A coluna 7 mostra os valores de y_i truncados em quatro algarismos. A coluna 8 mostra os valores de y_i arredondados em quatro algarismos. A coluna 9 mostra os valores de y_i calculados utilizando $y(x_{i-1})$. A coluna 10 mostra o erro local (e_1) onde $e_1 = y_i - y(x_i)$. A coluna 11 mostra o erro máximo ($e_{máx}$) onde:

$$e_{máx} = \frac{h^2}{2!} e^{x_i+1} \text{ , sendo } x_{i+1}=x \qquad \text{(C.9)}$$

Tabela C.1 – Retirada de *Carnahan, Brice; Luther, H. A.; Wilkes, J. O.;1969*

$Y(0) = 0$							$h = 0.1$			
i (1)	x_i (2)	y_i (3)	$f(x_i, y_i)$ (4)	$y(x_i)$ (5)	e (6)	y_i (7)	y_i (8)	y_i (9)	e_1 (10)	$e_{máx}$ (11)
0	0.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	—	—
1	0.1	0.0000	0.1000	0.0052	–0.0052	0.0000	0.0000	0.0000	–0.0052	0.0055
2	0.2	0.1000	0.2100	0.0214	–0.0114	0.0100	0.0100	0.0157	–0.0057	0.0061
3	0.3	0.0310	0.3310	0.0499	–0.0189	0.0310	0.0310	0.0435	–0.0064	0.0067
4	0.4	0.0641	0.4641	0.0918	–0.0277	0.0641	0.0641	0.0849	–0.0069	0.0067
5	0.5	0.11051	0.61051	0.1487	–0.0382	0.1105	0.1105	0.1410	–0.0077	0.0075
6	0.6	0.171561	0.771561	0.2221	–0.0505	0.1715	0.1716	0.2136	–0.0085	0.0082
7	0.7	0.2487171	0.9487171	0.3138	–0.0651	0.2486	0.2488	0.3043	–0.0095	0.0091
8	0.8	0.34358881	1.1435881	0.4255	–0.0819	0.3434	0.3437	0.4152	–0.0103	0.0101
9	0.9	0.457947691	1.357947691	0.5596	–0.1017	0.4577	0.4581	0.5480	–0.0116	0.0111
10	1.0	0.5937424601	1.5937424601	0.7183	–0.1246	0.5934	0.5939	0.7056	–0.0127	0.0123

Para um melhor entendimento foi feita uma verificação dos cálculos com o auxílio do Excel 97. Os resultados obtidos estão apresentados na Tabela C.2 a seguir:

Tabela C.2 – Resultados, utilizando o Método de Euler, obtidos com o auxílio do Microsoft Excel 97

$y(0) = 0$							$h = 0.1$			
i (1)	X_i (2)	y_i (3)	$f(x_i, y_i)$ (4)	$y(x_i)$ (5)	e (6)	y_i (7)	y_i (8)	y_i (9)	e_1 (10)	$e_{máx}$ (11)
0	0	0	0	0	0	0	0	0	0	0
1	0,1	0	0,1	0,0052	-0,0052	0	0	0	-0,0052	0,0055
2	0,2	0,01	0,21	0,0214	-0,0114	0,01	0,01	0,0157	-0,0057	0,0061
3	0,3	0,031	0,331	0,0499	-0,0189	0,031	0,031	0,0435	-0,0064	0,0067
4	0,4	0,0641	0,4641	0,0918	-0,0277	0,0641	0,0641	0,0849	-0,0069	0,0075
5	0,5	0,11051	0,61051	0,1487	-0,0382	0,1105	0,1105	0,141	-0,0077	0,0082
6	0,6	0,171561	0,771561	0,2221	-0,0505	0,1715	0,1716	0,2136	-0,0084	0,0091
7	0,7	0,2487171	0,9487171	0,3138	-0,0651	0,2486	0,2488	0,3043	-0,0095	0,0101
8	0,8	0,34358881	1,14358881	0,4255	-0,0819	0,3434	0,3437	0,4152	-0,0103	0,0111
9	0,9	0,457947691	1,357947691	0,5596	-0,1017	0,4577	0,4581	0,5481	-0,0115	0,0123
10	1	0,59374246	1,59374246	0,7183	-0,1246	0,5934	0,5939	0,7056	-0,0127	0,0136

C.3 – MÉTODOS DE RUNGE – KUTTA

É possível desenvolver um procedimento de um passo que envolve somente cálculos de derivadas de primeira ordem que é equivalente em exatidão aos resultados obtidos pelas fórmulas de Taylor de ordens elevadas. Esse procedimento é chamado Método de Runge – Kutta. Neste procedimento podem ser realizadas aproximações de segunda, terceira, quarta, ou seja, *m* ordem (isto é, aproximações equivalentes em exatidão a expansão da Série de Taylor). Para ser utilizado é necessário uma estimação de *f*(*x, y*) em dois, três , e quatro valores, respectivamente, de *x* no intervalo $x_i \leq x \leq x_{i+1}$.

Para ordens *m*, onde *m* é maior que quatro, é necessário uma estimação de *f*(*x, y*) em mais que *m* pontos.

Todos os métodos de Runge – Kutta têm a forma:

$$y_{i+1} = y_i + h\phi(x_i, y_i, h) \qquad \text{(C.10)}$$

Onde ϕ é denominado incremento da função, uma aproximação adequada para *f*(*x, y*) no intervalo $x_i \leq x \leq x_{i+1}$.

O Método de Runge – Kutta de terceira ordem é dado por:

$$y_{i+1} = y_i + \frac{h}{6}\left(k_1 + 4k_2 + k_3\right), \qquad \text{(C.11)}$$

$$k_1 = f\left(x_i, y_i\right)$$
$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1\right)$$
$$k_3 = f\left(x_i + h, y_i + 2hk_2 - hk_1\right)$$

A seguir está o Método de Runge – Kutta de quarta ordem:

$$y_{i+1} = y_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{(C.12)}$$
$$k_1 = f\left(x_i, y_i\right)$$
$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1\right)$$
$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2\right)$$
$$k_4 = f\left(x_i + h, y_i + hk_3\right)$$

Outro Método de quarta ordem também escrito por Kutta é:

$$y_{i+1} = y_i + \frac{h}{8}\left(k_1 + 3k_2 + 3k_3 + k_4\right), \qquad \text{(C.13)}$$
$$k_1 = f\left(x_i, y_i\right)$$
$$k_2 = f\left(x_i + \frac{1}{3}h, y_i + \frac{1}{3}k_1\right)$$
$$k_3 = f\left(x_i + \frac{2}{3}h, y_i - \frac{1}{3}hk_1 + hk_2\right)$$
$$k_4 = f\left(x_i + h, y_i + hk_1 - hk_2 + hk_3\right)$$

Ex: Consideremos a equação diferencial anterior:

$$f(x, y) = \frac{dy}{dx} = x + y \qquad \text{(C.6)}$$

Dada a condição inicial $x_0 = 0$, $y(x_0) = y_0 = 0$ a solução analítica é:

$$y = e^x - x - 1 \qquad \text{(C.7)}$$

Foram feitos os cálculos, com a ajuda do Micrsosoft Excel 97, utilizando o Método de Runge – Kutta, de terceira e quarta ordem, com um passo $h = 0.1$ e um limite superior de integração $x_{10} = 1.0$. Os resultados estão exibidos na Tabela C.3. As colunas 3 e 4 contém os valores de y_i calculados pelo Método de Runge – Kutta de terceira ordem (C.11), sendo a quarta coluna os valores arredondados em quatro algarismos. As colunas 5 e 6 contém os valores de y_i calculados pelo Método de Runge – Kutta de quarta ordem (C.12), sendo a sexta coluna os valores arredondados em quatro algarismos. As colunas 7 e 8 contém os valores de y_i calculados pelo outro Método de Runge – Kutta de quarta ordem (C.13), sendo a oitava coluna os valores arredondados em quatro algarismos. As colunas 9 e 10 contém os valores de y_i calculados analiticamente (C.7), sendo a décima coluna os valores arredondados em quatro algarismos.

Tabela C.3 – Resultados, utilizando o Método de Runge – Kutta, obtidos com o auxílio do Microsoft Excel

dy/dx = f(x, y)= x + y									
y(0) = 0								***H=0.1***	
***i* (1)**	***x_i* (2)**	***Y_i* 3^a *ordem* (3)**	***y_i* arred. 3^a *ordem* (4)**	***y_i* 4^a *ordem* (5)**	***y_i* arred. 4^a *ordem* (6)**	**y_i** 4^a *ordem* (7)	**y_i** *arred.* 4^a *ordem* (8)	***y_i* analítico (9)**	***y_i* arred. *analítico* (10)**
0	0	0	0	0	0	0	0	0	0
1	0,1	0,005166667	0,0052	0,005170833	0,0052	0,005170833	0,0052	0,005170918	0,0052
2	0,2	0,021393361	0,0214	0,021402571	0,0214	0,021402571	0,0214	0,021402758	0,0214
3	0,3	0,04984323	0,0499	0,049858497	0,0499	0,049858497	0,0499	0,049858808	0,0499
4	0,4	0,091801743	0,0919	0,09182424	0,0919	0,09182424	0,0919	0,091824698	0,0918
5	0,5	0,148689559	0,1488	0,148720639	0,1488	0,148720639	0,1488	0,148721271	0,1487
6	0,6	0,222076744	0,2222	0,222117962	0,2222	0,222117962	0,2222	0,2221188	0,2221
7	0,7	0,313698482	0,3138	0,313751627	0,3138	0,313751627	0,3138	0,313752707	0,3138
8	0,8	0,425472439	0,4256	0,425539563	0,4256	0,425539563	0,4256	0,425540928	0,4255
9	0,9	0,559517957	0,5597	0,559601414	0,5597	0,559601414	0,5597	0,559603111	0,5596
10	1	0,718177262	0,7184	0,718279744	0,7184	0,718279744	0,7184	0,718281828	0,7183

C.4 – CONCLUSÃO

Podemos notar que no Método de Euler o erro aparece a partir da primeira casa decimal; no entanto, no Método de Runge – Kutta o erro aparece apenas a partir da quarta casa decimal, para o mesmo número de iterações. Conclui-se então que o método de Runge – Kutta, apesar de requerer mais trabalho, possui uma melhor precisão.

APÊNDICE D – ESTUDO DO MATLAB

D.1 – INTRODUÇÃO

Realizamos um estudo do MATLAB para auxiliar na programação da propagação de detritos espaciais, utilizando como principal referência *Prodenge; Hanselman,Duane; Littlefield*. Este estudo nos possibilitou um melhor entendimento sobre os programas desenvolvidos anteriormente, e na programação dos posteriores.

Foram abordados neste estudo operações com matrizes, gráficos no R^2, funções matemáticas elementares, funções trigonométricas, operações aritméticas com números complexos e o comando de loop *for*.

D.2 – OPERAÇÕES COM MATRIZES

O estudo de operações com matrizes incluiu: matrizes transpostas, multiplicação de matrizes, matriz power, determinante de uma matriz e matriz inversa. A seguir estão os comandos e um exemplo de cada uma.

Matrizes Transpostas: No MATLAB a matriz transposta é denotada por A'.
Ex: Considere a matriz A

$$A = \begin{bmatrix} 2 & 5 & 6 \\ 3 & 11 & 9 \end{bmatrix}$$

Utilizando o MATLAB:

```
>> A=[2 5 6;3 9 11]

A =

    2    5    6
    3    9   11

>> A'

ans =

    2    3
    5    9
    6   11
```

Multiplicação de Matrizes: No MATLAB podem ser usados os seguintes comandos:
Ex: Sejam B e C respectivamente:

$$B = \begin{bmatrix} 2 & 5 & 1 \\ 0 & 3 & -1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 2 \\ -1 & 4 & -2 \\ 5 & 2 & 1 \end{bmatrix}$$

Utilizando o MATLAB:

```
>> B=[2 5 1;0 3 -1]

B =

    2    5    1
    0    3   -1

>> C=[1 0 2;-1 4 -2;5 2 1]

C =

    1    0    2
   -1    4   -2
    5    2    1

>> B*C

ans =

    2   22   -5
   -8   10   -7
```

Matriz Power: Quando se tem uma matriz quadrada e se deseja calcular D*D, usa-se a operação D^2. Lembrando que D^4 equivale a D*D*D*D.
Ex:

$$D = \begin{bmatrix} 2 & 3 \\ 7 & 11 \end{bmatrix}$$

Utilizando o MATLAB:

```
>> D=[2 3;7 11]

D =

    2    3
```

```
    7    11

>> D^3

ans =

         323         504
        1176        1835

>>
```

Determinante: No MATLAB, o comando utilizado para se achar o determinante de uma matriz E é det(E).
Ex:

$$E = \begin{bmatrix} 2 & 3 \\ 9 & 5 \end{bmatrix}$$

Utilizando o MATLAB:

```
>> E=[2 3;9 5]

E =

    2    3
    9    5

>> det(E)

ans =

   -17
```

Matriz Inversa: No MATLAB, para obtermos uma matriz inversa devemos fornecer a matriz original F e executar o comando inv(F).

$$F = \begin{bmatrix} 8 & 13 \\ 17 & 5 \end{bmatrix}$$

Utilizando o MATLAB:

```
>> F=[8 13;17 5]

F =

     8    13
    17     5

>> inv(F)

ans =

   -0.0276    0.0718
    0.0939   -0.0442
```

D.3 – GRÁFICOS NO R^2

Podemos usar o MATLAB para plotar gráficos. A seguir está um exemplo de como gerar um simples gráfico *x – y* de dados armazenados em dois vetores.

Ex: Plotaremos os dados de temperatura a seguir coletados em uma experiência física (tabela retirada de *Prodenge*).

Tabela D.1 –Retirada de *Prodenge*

Tempo (s)	**Temperatura (^{o}C)**
0	54.2
1	58.5
2	63.8
3	64.2
4	67.3
5	71.5
6	88.5
7	90.1
8	90.6
9	89.5
10	90.4

Supomos também que os dados relativos ao tempo estejam armazenados em um vetor denominado *x*, e que os relativos à temperatura estejam armazenados em um vetor denominado *y*. Para plotar estes pontos, simplesmente usamos o comando *plot* , onde *x e y* são vetores – linha ou vetores – coluna.

O gráfico é gerado automaticamente. Para incluir unidades, título e uma breve descrição. Utilizamos os seguintes comandos:
title – Adiciona um título ao gráfico.

xlabel – Inclui uma descrição na direção do eixo-*x*.
xlabel – Inclui uma descrição na direção do eixo-*y*.
grid – Adiciona linhas de grade ao gráfico.
whitebg – Muda a cor de fundo do gráfico para branco.

Se utilizarmos após cada comando três pontos o MATLAB executa os passos de uma só vez.

Assim, utilizando o MATLAB:

```
>> x=[0 1 2 3 4 5 6 7 8 9 10];

>> y=[54.2 58.5 63.8 64.2 67.3 71.5 88.5 90.1 90.6 89.5 90.4];

>> plot(x,y) ...

>> title('Experiência de Física')...

>> xlabel('Tempo (s)')...

>> ylabel('Temperatura (Graus Celsius)')...

>> grid...

>> whitebg...
```

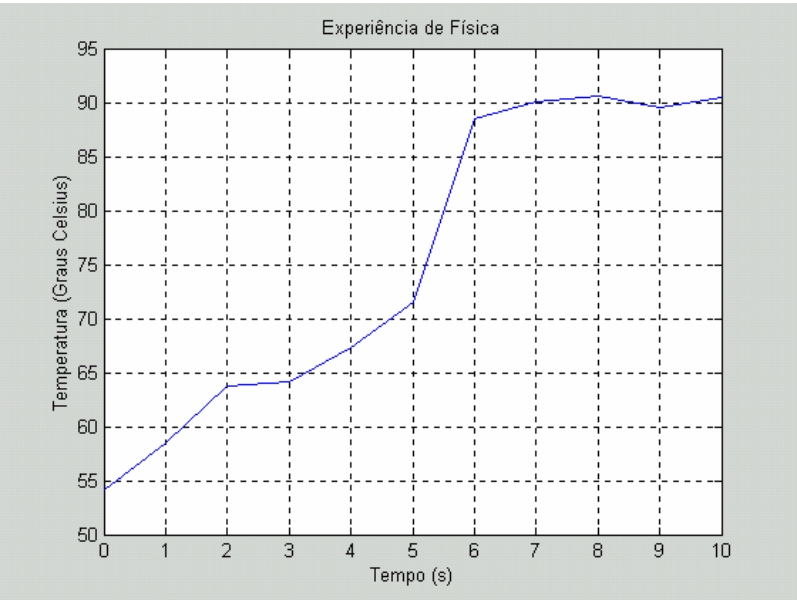

Figura D.1 – Experiência de Física

D.4 – OPERAÇÕES ARITMÉTICAS COM NÚMEROS COMPLEXOS

Os comandos do MATLAB reconhecem os números complexos usando *i* e *j* para representar √-1.

O comando a seguir define uma variável complexa:

x = 1 - 0.5*i

Quando executamos operações entre dois complexos, o MATLAB automaticamente executa os cálculos necessários. Se uma operação for entre um número real e um complexo, o MATLAB supõe que a parte imaginária do número real é igual a zero.

O MATLAB inclui várias funções que são específicas aos números complexos, tais como:

real (x) – Calcula a parte real do número complexo x.
imag(x) – Calcula a parte imaginária do número complexo x.
conj(x) – Calcula o conjugado do número complexo x.
abs(x) – Calcula o módulo do número complexo x.
angle(x) – Calcula o ângulo usando o valor de atan2 (imag(x), real(x)), e portanto o ângulo está entre -π e π.

Com estas funções tornam mais fácil converter o complexo da forma polar para retangular.

Ex:

Utilizando o MATLAB:

```
>> y=3+4*i

y =

   3.0000 + 4.0000i

>> real(y)

ans =

    3

>> imag(y)

ans =

    4

>> conj(y)
```

```
ans =

   3.0000 - 4.0000i

>> abs(y)

ans =

     5

>> angle(y)

ans =

    0.9273
```

D.5 – COMANDO DE LOOP *FOR*

Um dos comandos do MATLAB para gerar loop é o comando *for*. O comando *for* tem a estrutura a seguir:

```
for variável = expressão
Grupo de comandos A
end
```

Ex:

```
>> for x=-5:0.01:5
y=x^2;
plot(x,y)
hold on
end
```

Obs: O comando hold on mantém as propriedades atuais dos comandos de tal forma que os gráficos subsequentes possam ser sobrepostos ao gráfico já existente.

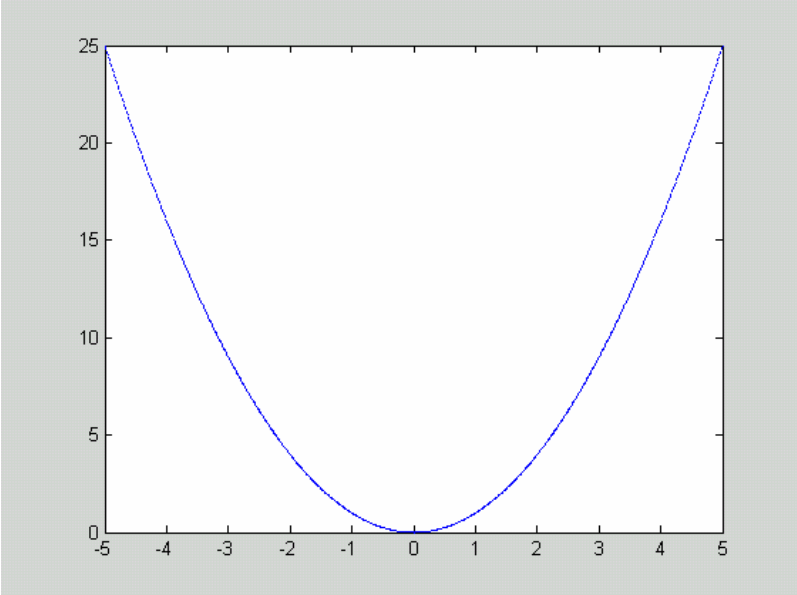

Figura D.2 – $y = x^2$

D.6 – EQUAÇÕES DIFERENCIAIS ORDINÁRIAS

É possível resolver equações diferenciais ordinárias numericamente, utilizando o MATLAB, este recurso é conveniente quando as equações não podem ser resolvidas facilmente de forma analítica.

Para resolver equações diferenciais utilizando o MATLAB é necessário reescrever as equações diferenciais de ordem superior em termos de um conjunto equivalente de equações diferenciais de primeira ordem.

O MATLAB contém dois comandos para calcular soluções numéricas para equações diferenciais ordinárias: *ode23* e *ode45*; o comando *ode23* usa o método de Runge - Kutta para equações diferenciais de segunda e terceira ordem; o comando *ode45* usa o método de Runge - Kutta para equações diferenciais de quarta e quinta ordem. Os comandos *ode23* e *ode45* possuem os mesmos tipos de argumentos.

Para utilizar estes comandos é necessário que escrevamos um arquivo M de função que retorna as equações diferenciais, de primeira ordem, anteriores e escrevê – las como um vetor coluna y, dados os valores iniciais (yo), e o intervalo de variação do tempo (intervalo). A seguir está o comando ode45:

```
ode45(@funcao, intervalo, yo);
```

Quando o as funções de EDO são usadas sem argumento de saída, elas geram um gráfico de evolução temporal do sistema. Para ter acessso aos dados, basta prover argumentos de saída:

[t,y]= ode45(@funcao, intervalo, yo);

onde t é um vetor coluna contendo os instantes de tempo em que a solução foi calculada e y é uma matriz com duas colunas. A primeira coluna de y corresponde a variável y(1) e a segunda, à variável y(2).

EXEMPLO 1: A Equação diferencial:

$$\frac{dy}{dx} = x + y \qquad \text{(D.1)}$$

Definindo novas variáveis:

$$y_1 = x \qquad \text{e} \qquad y_2 = y \qquad \text{(D.2)}$$

Então podemos dizer que:

$$y_1' = 1 \qquad \text{e} \qquad y_2' = y_1 + y_2 \qquad \text{(D.3)}$$

A seguir está o arquivo M de função:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function yponto=funcao(t,y)
% Equacao diferencial:
% y'=x+y
% Definido novas variaveis:
% y(1)=x e y(2)=y
%entao:y(1)'=1 e, pois dx/dx=1
%y(2)'= y(1)+y(2)
%yponto=[y(1)',y(2)']

yponto=[1;y(1)+y(2)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Utilizando o MATLAB:

```
>> intervalo=[0 1];
>> yo=[0;0];
```

```
>> ode45(@funcao,intervalo,yo);
```

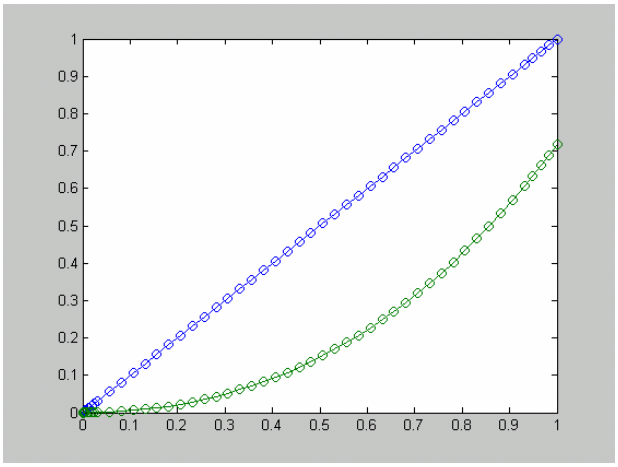

Figura D.3 – Resolução da Equação Diferencial

Plotando o gráfico de y(1) no tempo, e y(2) no tempo:

Utilizando o MATLAB:

```
>> intervalo=[0 1];
>> yo=[0;0];
>> [t,y]=ode45(@funcao,intervalo,yo);
>> figure
>> plot(t,y(:,1),'b')
title('y(1)')
>> figure
>> plot(t,y(:,2),'g')
title('y(2)')
>>
```

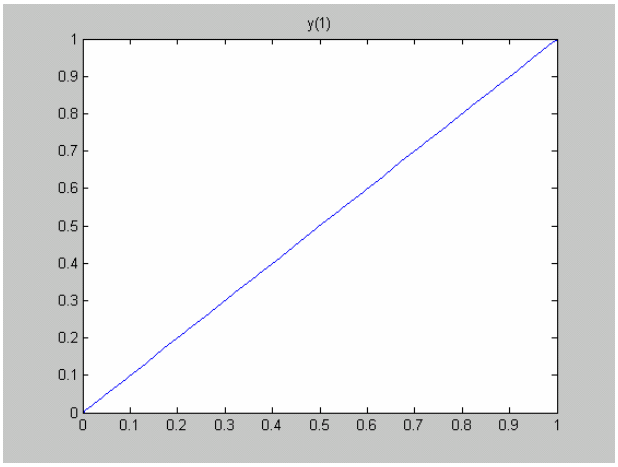

Figura D.4 – y(1) = x

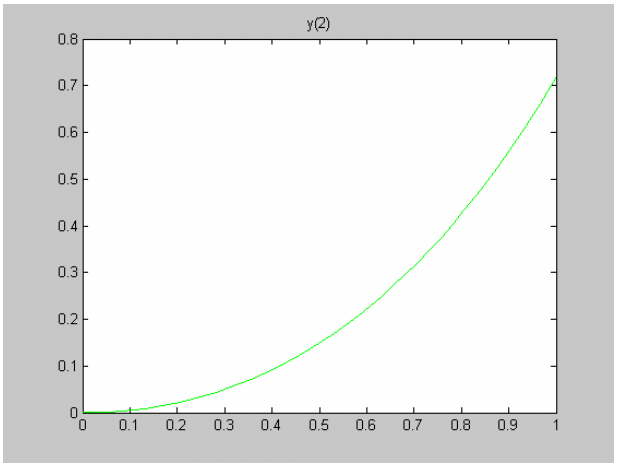

Figura D.5 – y(2) = y

EXEMPLO 2: A equação diferencial de Van der Pol.

$$\frac{d^2x}{dt^2} - \mu\left(1 - x^2\right)\frac{dx}{dt} + x = 0 \qquad \text{(D.4)}$$

Definindo novas variáveis:

$$y_1 = x \qquad \text{e} \qquad y_2 = \frac{dx}{dt} \qquad \text{(D.5)}$$

A equação diferencial de Van der Pol pode ser reescrita como:

$$\frac{d^2x}{dt^2} = \mu\left(1 - x^2\right)\frac{dx}{dt} - x \qquad \text{(D.6)}$$

Substituindo a equação D.5 na equação D.6, podemos dizer que:

$$\frac{dy_2}{dt} = \mu\left(1 - y_1^2\right)y_2 - y_1 = 0 \qquad \text{(D.7)}$$

A seguir está o arquivo M de função:

```
%%%%%%%%%%%%%%%%%%%%

function yponto=vonder(t,y)
% Equacao de Van der Pol:
% x''-mu*(1-x^2)*x'+x=0
% Definido novas variaveis:
% y(1)=x e y(2)=x'
%entao:y(1)'=y(2) e
%y(2)'= mu*(1-(y(1)).^2)*y(2)-y(1)

mu=2;
yponto=[y(2);mu*(1-(y(1)).^2)*y(2)-y(1)];

%%%%%%%%%%%%%%%%%%%%
```

Utilizando o MATLAB:

```
>> intervalo=[0 30];
>> yo=[1;0];
>> ode45(@vonder,intervalo,yo)
```

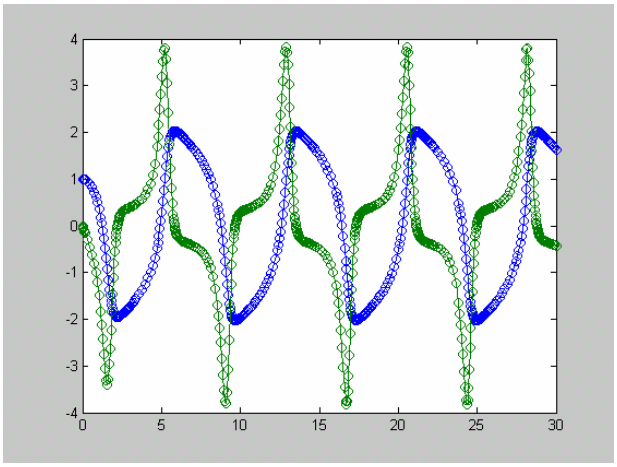

Figura D.6 – Resolução da Equação de Van der Pol

Plotando o gráfico de y(1) no tempo, e y(2) no tempo:

Utilizando o MATLAB:

```
>> intervalo=[0 30];
>> yo=[1;0];
>> [t,y]=ode45(@vonder,intervalo,yo);
>> figure
>> plot(t,y(:,1))
title('y(1)')
>> figure
>> plot(t,y(:,2),'g')
title('y(2)')
>>
```

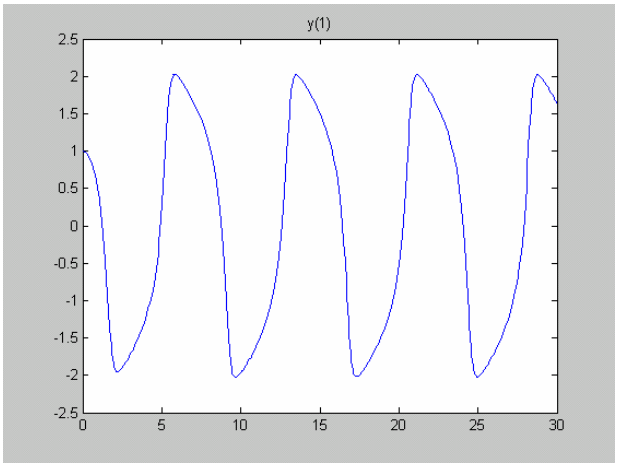

Figura D.7 – y(1) = x

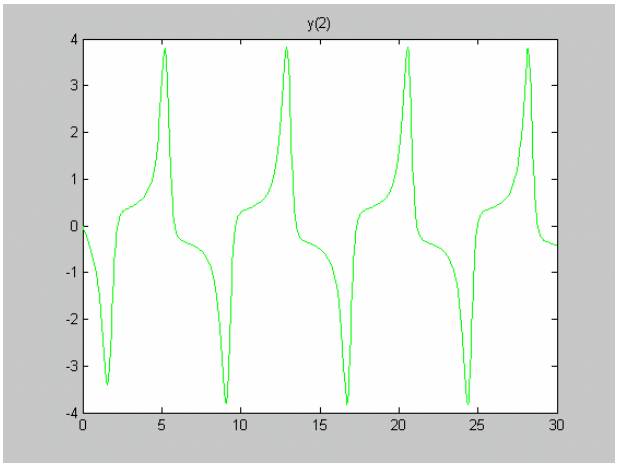

Figura D.8 – y(2) = x’

D.7 – CONCLUSÃO

O estudo do MATLAB foi essencial na programação da propagação de detritos espaciais. Pois foi ele que nos possibilitou um melhor entendimento sobre os programas desenvolvidos anteriormente, nos proporcionando uma melhor base para a programação dos posteriores.