GSDAE : Código para Resolver EAD com Singularidades

Antonio Castelo Renato César da Silva
Departamento de Ciências de Computação e Estatística
Instituto de Ciências Matemáticas de São Carlos
Universidade de São Paulo (USP)
São Carlos - SP
Brazil
e-mail: castelo@icmsc.sc.usp.br
e-mail: renato@icmsc.sc.usp.br

Geovan Tavares
Departamento de Matemática
Pontifícia Universidade Católica (PUC-Rio)
R. Marquês de São Vicente, 225
22453-900 Rio de Janeiro - RJ
Brazil
e-mail: geovan@mat.puc-rio.br

Abstract

In this work we introduce the GSDAE solver for implicit differential equations exhibiting singularities. Our approach is a geometrical one and we use the concept of contact structure on a manifold associated with the differential equation [6]. We indicate how the method was implemented using continuation and BDF methods. The fixed leading coefficient technique and the strategies for stepsize and order selection and error control used in the GSDAE are similar to the ones implemented in the DASSL solver [2].

1 Introdução

Uma Equação Diferencial Implícita (EDI) de ordem k tem a seguinte forma :

$$F(X, Y(X), Y'(X), \ldots, Y^{(k)}(X)) = 0 \tag{1}$$

onde $F : \Omega \to I\!R^n$ é uma aplicação suficientemente diferenciável no conjunto aberto e conexo $\Omega \subset I\!R^{(k+1)n+1}$.

Esta equação, também denominada Equação Algébrico-Diferencial (EAD), tem sido sujeito de estudo em vários artigos nos últimos anos. Estes artigos descrevem sobre aspéctos teóricos, métodos numéricos e aplicações. Para referências gerais veja Brenan, Campbell e Petzold [2] e Hairer e Wanner [12]. Os métodos numéricos mais explorados são para EAD de ordem 0 e 1 com índice fixo, e entre eles os que mais se destacam são os métodos Backward Differenciation Formula (BDF) e Runge-Kutta Implicito (RKI).

Sobre o ponto de vista geométrico, Rheinboldt [19] deduziu aproximações para reduzir localmente uma EAD em uma Equação Diferencial Ordinária (EDO) sobre variedades. Estas aproximações foram exploradas posteriormente em Rabier e Rheinboldt [18].

EAD com singularidades não tem sido muito tratada na literatura. Wasow [24] usou expansão assintótica para estudar EAD linear no domínio complexo onde apresentam singularidades. Para uma única equação, Spence [13], combinou informações geométricas com métodos de continuação para determinar pontos singulares. A equação geral, para o caso de dobras simples, foi tratada em Rabier [17]. Recentemente, von Sosen [22] tratou o problema de bifurcação para EAD na forma semi-explícita.

O pioneiro no uso de estruturas de contacto no estudo de EAD foi Darboux [7] e posteriormente estas estruturas foram revistas por Thom [23]. Estas aproximações foram exploradas para uma única equação por Lak Dara [8] e Davydov [9] que apresentaram várias formas normais. Para mais referências veja Arnold [1].

Em Castelo, Freitas e Tavares [5] e em Freitas e Tavares [11] foi utilizado o método simplicial para aproximar soluções de EAD empregando estruturas de contacto. Posteriormente, Castelo e Tavares [6] utilizaram métodos de continuação do tipo preditor-corretor (Euler explícito - Euler implícito) juntamente com as estruturas de contacto para resolver EAD de índice 0 ou de índice 1 na forma semi-implícita.

Este trabalho apresenta o código GSDAE, desenvolvido em linguagem C, para obter aproximações de soluções clássicas ou soluções gerais de EAD de ordem qualquer (incluindo ordem 0, que é uma equação puramente algébrica) na forma implícita ou na forma semi-implícita. Ele utiliza técnicas dos métodos de continuação e o método BDF. As estratégias e métodos utilizados no GSDAE são baseadas nas do código DASSL [2] para predição, correção, contrôle do tamanho do passo e ordem.

Também apresenta-se neste trabalho as rotinas de interface, os tipos de variaveis do GSDAE e como utilizar o código, além de alguns exemplos de utilização.

Na seção 2 apresenta-se um resumo da teoria sobre EAD com singualridades descrita em [6], na seção 3 métodos numéricos para EAD com singularidades, na seção 4 os métodos e estratégias utilizadas no GSDAE, na seção 5 a descrição do GSDAE contendo os tipos de

variáveis, as rotinas de interface, variáveis, as rotinas ALLOCPAR, FREEPAR, CSDAE, GSDAE, STATUS, STATISTICS e as rotinas definidas pelo usuário. Finalmente na seção 6 apresenta-se alguns exemplos de utilização do código.

2 Equações Algébrico-Diferenciais com Singularidades

Definição 2.1 Uma aplicação $Y : \mathcal{I} \to I\!R^n$ é denominada solução clássica da equação (1) se $Y \in C^k(\mathcal{I})$ e satisfaz (1).

Definição 2.2 A *imagem* de uma aplicação $c : I \to I\!R^{(k+1)n+1}$ é denominada solução geral da equação (1) se $c \in C^1(I)$ e satisfaz

$$\left\{\begin{array}{lcl} F(c(s)) & = & 0 \\ y_1(s)x'(s) - y_0'(s) & = & 0 \\ y_2(s)x'(s) - y_1'(s) & = & 0 \\ & \vdots & \\ y_k(s)x'(s) - y_{k-1}'(s) & = & 0 \end{array}\right. \tag{2}$$

para todo $s \in I$, onde $c(s) = (x(s), y_0(s), y_1(s), \ldots, y_k(s))$.

Lema 2.1 Seja uma solução geral de (1) dada por $c : I \to I\!R^{(k+1)n+1}$. Se $x'(s) \neq 0$ em um intervalo $\mathcal{I} \subset I$, então $y_0(s) = Y_0(x(s)), y_1(s) = \frac{d}{dx}Y_0(x(s)), \ldots, y_k(s) = \frac{d^k}{dx^k}Y_0(x(s))$ sobre $\mathcal{I}$ e Y_0 definido em $x(\mathcal{I})$ é uma solução clássica de (1).

O lema 2.1 diz que pode-se obter soluções clássicas a partir de soluções gerais de (1) através da projeção no plano x, y_0.

Definição 2.3 O problema de valor inicial para a equação (1) é definido por :

$$\left\{\begin{array}{c} F(X, Y(X), Y'(X), \ldots, Y^{(k)}(X)) = 0 \\ Y^{(i)}(x_0) = a_i, i = 0, 1, \ldots, k, \end{array}\right. \tag{3}$$

onde $F(x_0, a_0, a_1, \ldots, a_k) = 0$.

Definição 2.4 Uma ω-*estrutura* para a equação (1) é uma aplicação $\omega : \Omega \to I\!R^{kn((k+1)n+1)}$ definida por :

$$\omega(x, y_0, y_1, \ldots, y_k) = \left(\begin{array}{c} \omega_k(x, y_0, y_1, \ldots, y_k) \\ \omega_{k-1}(x, y_0, y_1, \ldots, y_k) \\ \vdots \\ \omega_2(x, y_0, y_1, \ldots, y_k) \\ \omega_1(x, y_0, y_1, \ldots, y_k) \end{array}\right) = \left(\begin{array}{ccccccc} y_k & 0 & 0 & \cdots & 0 & -I_n & 0 \\ y_{k-1} & 0 & 0 & \cdots & -I_n & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ y_2 & 0 & -I_n & \cdots & 0 & 0 & 0 \\ y_1 & -I_n & 0 & \cdots & 0 & 0 & 0 \end{array}\right).$$

Com esta ω-*estrutura* pode-se reescrever (2) na forma :

$$\begin{cases} F(c(s)) & = 0 \\ \omega(c(s))c'(s) & = 0 \end{cases} \tag{4}$$

que será denominado *sistema diferencial* associado a (1).

O problema de valor inicial para o sistema diferencial (4) é naturalmente definido por :

$$\begin{cases} F(c(s)) & = 0 \\ \omega(c(s))c'(s) & = 0 \\ c(0) & = c_0 = (x_0, a_0, a_1, \ldots, a_k), \end{cases} \tag{5}$$

com $F(c_0) = 0$.

Diferenciando a primeira equação de (5) com respeito a s e definindo

$$A(c(s)) = \begin{pmatrix} DF(c(s)) \\ \omega(c(s)) \end{pmatrix}, \tag{6}$$

temos o seguinte problema de valor inicial :

$$\begin{cases} A(c(s))c'(s) & = 0 \\ c(0) & = c_0 = (x_0, a_0, a_1, \ldots, a_k), \end{cases} \tag{7}$$

com $F(c_0) = 0$.

Teorema 2.1 Se DF é Lipschitz em uma vizinhança aberta de c_0 e $A(c_0)$ tem posto máximo, então existe $r > 0$ tal que o problema de valor inicial (5) tem uma única solução (no sentido de que o traçé único) sobre $B = \{u; \parallel u - c_0 \parallel \leq r\}$.

Corolário 2.1 Se DF é Lipschitz em uma vizinhança aberta de $(x_0, a_0, a_1, \ldots, a_k)$ e $A(x_0, a_0, a_1, \ldots, a_k)$ tem posto máximo, então existe $r > 0$ tal que o problema de valor inicial (3) tem uma única solução geral em $B = \{(X, Y_0, \ldots, Y_k); \parallel (Y, Y_0, \ldots, Y_k) - (x_0, a_0, \ldots, a_k) \parallel \leq r\}$.

Definição 2.5 Um ponto P_0 é uma singularidade da equação (1) se $F(P_0) = 0$ e $F_{Y^{(k)}}$ é singular em P_0 mas não em uma vizinhança de P_0.

Proposição 2.1 Seja uma solução geral de (1) dada por $c : I \to I\!R^{nk+1}$, $c(s) = (x(s), y_0(s), \ldots, y_k(s))$ com $c'(s_0) \neq 0$ e DF Lipchitz em uma vizinhança de $c(s_0)$.

1) Se $x'(s_0) = 0$, então $y_0'(s_0) = 0, \ldots, y_{k-1}'(s_0) = 0$, $y_k'(s_0) \neq 0$ e $F_{y_k}(c(s_0))$ é singular.

2) Se $A(c(s_0))$ tem posto máximo e $F_{y_k}(c(s_0))$ é singular, então $x'(s_0) = 0$, $y_0'(s_0) = 0,\ldots,y_{k-1}'(s_0) = 0$, $y_k'(s_0) \neq 0$ e $F_{y_k}(c(s_0))$ tem posto $n-1$.

Segue desta proposição que $x(s) = x(s_0)+x''(s_0)\frac{(s-s_0)^2}{2}+\cdots x^{(p)}(s_0)\frac{(s-s_0)^p}{p!}+O((s-s_0)^{p+1})$. Se $x^{(i)}(s_0) = 0$, $i = 2,\ldots,p-1$ e $x^{(p)}(s_0) \neq 0$, então $c(s)$ tem uma singularidade de ordem p em s_0. Se $p = 2$ então $c(s)$ tem uma dobra simples como singularidade. Por mudanças de coordenadas pode-se tomar $x(s) - x(s_0) = s^2$, portanto no espaço $x, y_0, \ldots, y_{k-1}$ tem-se duas soluções, uma chegando e outra saindo do ponto $(x(s_0), y_0(s_0), \ldots, y_{k-1}(s_0))$.

Se $F_{y_k}(c_0)$ tem posto menor que $n-1$, $A(c_0)$ não pode ter posto máximo. Caso $F_{y_k}(c_0)$ tenha posto constante $r < n-1$ em uma vizinhaça de c_0, pode-se aplicar mudanças de variáveis no domínio e contra-domínio de F de forma a escrever (1) na forma semi-implícita:

$$\left\{ \begin{array}{ll} f(X, Z(X), W(X), \ldots, Z^{(k-1)}(X), W^{(k-1)}(X), Z^{(k)}(X)) & = 0 \\ g(X, Z(X), W(X), \ldots, Z^{(k-1)}(X), W^{(k-1)}(X)) & = 0 \end{array} \right. \tag{8}$$

onde $f : \Omega \to I\!R^r$, $g : \Omega \to I\!R^{n-r}$ e $\Omega \subset I\!R^{kn+r+1}$.

Resultados análogos ao vistos acima são válidos para EAD na forma semi-implícita.

3 Métodos Numéricos para EAD com Singularidades

Uma possibilidade para obter-se aproximações de uma solução geral de (5) é resolver numéricamente o problema de valor inicial equivalente

$$\left\{ \begin{array}{ll} c'(s) & = \tau(A(c(s))) \\ c(0) & = c_0, \end{array} \right. \tag{9}$$

onde $\tau(A) \in I\!R^{n+1}$ é tal que $A\tau(A) = 0$, $\| \tau(A) \| = 1$ e $\det \left(\begin{array}{c} A \\ \tau^t(A) \end{array} \right) > 0$.

Um método de passos multiplos linear de r passos para (9) é dado por :

$$\sum_{i=0}^{r} \alpha_i c_{n-i} - h\beta_i \tau(A(c_{n-i})) = 0 \tag{10}$$

Tem-se que $\tau(A) = \pm\, z$, onde z é a ultima coluna da matriz Q da decomposição $A^t = Q \left(\begin{array}{c} R \\ 0^t \end{array} \right)$.

O custo desta avaliação pode ser diminuido observando a expressão de ω, de onde tira-se que

$$\begin{array}{rcl} y_1(s)\tau_x - \tau_{y_0} = 0 & \Rightarrow & \tau_{y_0} = y_1(s)\tau_x \\ y_2(s)\tau_x - \tau_{y_1} = 0 & \Rightarrow & \tau_{y_1} = y_2(s)\tau_x \\ \vdots & & \\ y_k(s)\tau_x - \tau_{y_{k-1}} = 0 & \Rightarrow & \tau_{y_{k-1}} = y_k(s)\tau_x \end{array} \tag{11}$$

onde $\tau = (\tau_x, \tau_{y_0}, \ldots, \tau_{y_k})$. Substituindo em $DF(c(s))\tau = 0$, segue que

$$B(c(s))\psi = \begin{pmatrix} F_x + F_{y_0}y_1 + \cdots + F_{y_{k-1}}y_k & F_{y_k} \end{pmatrix} \begin{pmatrix} \tau_x \\ \tau_{y_k} \end{pmatrix} = 0.$$

De maneira análoga à avaliação de τ, pode-se obter ψ usando a decomposição de QR de B^t, então usando as relações em (11), obtem-se τ.

O método numérico (10) pode não ser muito estável para algumas EAD's, mas pode ser usado como um método preditor e/ou inicializador.

Para corrigir, pode-se usar o problema de valor inicial (5) com um método corretor do tipo BDF (Backward Diferentiation Formulas)

$$\begin{cases} F(c_n) = 0 \\ \omega(c_n)\frac{\rho c_n}{h} = 0 \end{cases} \tag{12}$$

onde $\rho c_n = \sum_{i=0}^{r} \gamma_i c_{n-i}$. Para obter c_n, é usado um método de Newton para sistemas indeterminados.

O método de Newton modificado para sistemas indeterminados é dado por

$$c_n^{i+1} = c_n^i - \alpha DG^+(c_n^0)G(c_n^i)$$

onde

$$G(c_n) = \begin{pmatrix} F(c_n) \\ \omega(c_n)\rho c_n \end{pmatrix}$$

A matriz $DG(c_n^0)$ tem dimensão $(k+1)n + 1 \times (k+1)n$ e ela tem uma estrutura de esparcidade especial, mas esta estrutura não é aprovieitada no cálculo de $DG^+(c_n^0)$.

Outra alternativa é considerar uma parametrização particular para a solução geral, como por exemplo, parametrização por comprimento de arco, que implica em acrescentar uma linha ao sistema (5)

$$\begin{cases} F(c(s)) = 0 \\ \omega(c(s))c'(s) = 0 \\ \| c'(s) \|^2 - 1 = 0 \\ c(0) = c_0 \end{cases} \tag{13}$$

Assim, para este sistema, o método corretor do tipo BDF é dado por

$$\begin{cases} F(c_n) & = 0 \\ \omega(c_n)\rho c_n & = 0 \\ \| \rho c_n \|^2 - h^2 & = 0 \end{cases} \tag{14}$$

e para obter c_n, é usado um método de Newton modificado

$$c_n^{i+1} = c_n^i - \alpha DG^{-1}(c_n^0)G(c_n^i)$$

onde

$$G(c_n) = \begin{pmatrix} F(c_n) \\ \omega(c_n)\rho c_n \\ \| \rho c_n \|^2 - h^2 \end{pmatrix}$$

Embora este sistema contenha uma linha a mais, podemos aproveitar a estrutura de esparcidade no cálculo da decomposição da matriz $DG(c_n^0)$ apenas reordenando as variáveis ao escrever G, isto é, fazer $c(s) = (y_k(s), y_{k-1}(s), \ldots, y_1(s), y_0(s), x(s))$. Com esta reordenação, temos

$$DG(c_n^0) = \begin{pmatrix} F_{y_k}(c_n^0) & F_{y_{k-1}}(c_n^0) & F_{y_{k-2}}(c_n^0) & \cdots & F_{y_2}(c_n^0) & F_{y_1}(c_n^0) & F_{y_0}(c_n^0) & F_x(c_n^0) \\ \rho x_n^0 I & -\gamma_0 I & 0 & \cdots & 0 & 0 & 0 & \gamma_0 y_{k,n}^0 \\ 0 & \rho x_n^0 I & -\gamma_0 I & \cdots & 0 & 0 & 0 & \gamma_0 y_{k-1,n}^0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \rho x_n^0 I & -\gamma_0 I & 0 & \gamma_0 y_{2,n}^0 \\ 0 & 0 & 0 & \cdots & 0 & \rho x_n^0 I & -\gamma_0 I & \gamma_0 y_{1,n}^0 \\ 2\gamma_0\rho y_{k,n}^0 & 2\gamma_0\rho y_{k-1,n}^0 & 2\gamma_0\rho y_{k-2,n}^0 & \cdots & 2\gamma_0\rho y_{2,n}^0 & 2\gamma_0\rho y_{1,n}^0 & 2\gamma_0\rho y_{0,n}^0 & 2\gamma_0\rho x_n^0 \end{pmatrix}$$

4 Métodos e Estratégias utilizadas no GSDAE

O GSDAE é um programa desenvolvido em C para obter aproximações de soluções clássicas (rotina CSDAE) ou soluções gerais (rotina GSDAE) de EAD de ordem qualquer (incluindo ordem 0, que é uma equação puramente algébrica) na forma implícita ou na forma semi-implícita.

Ele utiliza o método de Euler explícito definido em (10) como inicializador, uma vez que $c'(0)$ não é dado como condição inicial, e o método BDF definido em (14).

As estratégias e métodos utilizados no GSDAE são baseadas nas do código DASSL para predição, correção, contrôle do tamanho do passo e ordem. Somente quando é detectada uma instabilidade, o método é reinicializado utilizando novamente o método (10), pois próximo a uma singularidade, o método de Euler explícito com passo pequeno proporciona uma aproximação melhor que o polinômio preditor.

A estratégia de integração com o método BDF é o de coeficiêntes principais fixos ([14],[2]) que é uma extensão o método BDF para passos variáveis. A idéia da estratégia de coeficiêntes princiapis fixos é de aplicar o método BDF com passo constante no polinômio que interpola a solução nos últimos k passos ($k = 1, \ldots, 5$).

Supondo ter os pontos $c_{n-k}, \ldots, c_n$ que são aproximações para a solução em $s_{n-k}, \ldots, s_n$, obtem-se o polinômio preditor

$$\chi_p(s_{n-i}) = c_{n-i} \quad i = 0, \ldots, k.$$

O polinômio preditor define uma primeira aproximação para o próximo ponto $c(s_{n+1})$ e sua derivada $c'(s_{n+1})$ ($s_{n+1} = s_n + h_{n+1}$), isto é,

$$\begin{array}{ll} c^0_{n+1} & = \chi_p(s_{n+1}) \\ c'^0_{n+1} & = \chi'_p(s_{n+1}). \end{array}$$

A solução para o próximo ponto é obtida utilizando o polinômio preditor e o método BDF com passo constante, isto é, considerando o polinômio corretor que interpola o polinônio preditor em k pontos igualmente espaçados

$$\chi_c(s_{n+1} - ih_{n+1}) = \chi_p(s_{n+1} - ih_{n+1}) \quad i = 1, \ldots, k.$$

e satisfaz o sistema

$$\left\{ \begin{array}{ll} F(\chi_c(s_{n+1})) & = 0 \\ \omega(\chi_c(s_{n+1}))\rho\chi_c(s_{n+1}) & = 0 \\ \parallel \rho\chi_c(s_{n+1}) \parallel^2 - h^2 & = 0. \end{array} \right.$$

Estes polinômios são representados utilizando diferenças divididas modificadas ([16],[20]) que são atualizadas a cada passo.

O critério para aceitação do passo, critério para mudança de ordem e do tamanho do passo é feita dependendo de estimativas dos termos do erro de truncamento ([20],[21]). Quanto a aceitação e amplitude do próximo passo, é utilizado o termo principal do erro de truncamanto, termo de ordem k, e para definir a ordem os termos do erro de truncamento de ordem $k-2$, $k-1$, k e $k+1$, de acordo com a monotocidade desta sequência. Quanto ao critério de parada para o método de Newton modificado e critério de aceitação do passo, é utilizada uma norma peso

$$\parallel v \parallel = \sqrt{\frac{\sum_1^m \left(\frac{v_i}{wt_i}\right)^2}{m}}$$

onde $wt_i = atol_i + |u_i| rtol_i$ define o vetor peso relativo ao vetor u, $atol$ o erro absoluto e $rtol$ o erro relativo desejado.

5 Descrição do GSDAE

O GSDAE pode ser utilizado para obter soluções clássicas, integrando até um valor para x definido pelo usuário ou até a primeira singularidade detectada. Caso o código detecte uma singularidade esta informação é retornada ao usuário. Ele também pode ser utilizado para integrar no parâmtro s (c é parametrizada por comprimento de arco) até um valor de s definido pelo usuário ou até uma singularidade.

5.1 Tipos de variáveis

Os tipos de variáveis utilizadas nos GSDAE estão definidas no módulo types.h. Para facilitar a mudança de tipo de ponto flutuante, é definido o tipo "real" que inicialmente é equivalente ao tipo "double". Para mudar este tipo para "float", se desejado, basta mudar uma única linha e recompilar o código. O GSDAE utiliza para vetores e matrizes alocados dinâmicamente. Vetores de inteiros são definidos por "vint", vetores de ponto flutuante por "vreal", matrizes de inteiros por "mint", matrizes de ponto flutuante por "mreal", matrizes tridimensionais de inteiros por "mmint" e matrizes tridimensionais de ponto flutuante por "mmreal".

Para o uso do GSDAE, o módulo que está a rotina que fará interface deve conter a inclusão:

#include "types.h"

5.2 Rotinas de Interface

O GSDAE possue 6 rotinas de interface com o usuário, elas são:

- ALLOCPAR : para alocar memoria;
- FREEPAR : para liberar memoria;
- GSDAE : para utilizar obter solucões gerais;
- CSDAE : para utilizar obter solucões clássicas;
- STATISTICS : para obeter dados sobre a execução do CSDAE ou GSDAE;
- STATUS : para obter a situação da integração das rotinas CSDAE ou GSDAE até o momento.

Para o uso do GSDAE, o módulo que está a rotina que fará interface deve conter a declaração das rotinas:

```
extern int GSDAE ( int, int, real, real, real, real, real *, real, real *, mreal, real,
        mreal, real, mreal, vreal, vint, vint );
extern int CSDAE ( int, int, real, real, real, real, real *, real *, real, mreal, real,
        mreal, real, mreal, vreal, vint, vint );
extern void ALLOCPAR ( int, int, mreal *, mreal *, mreal *, vreal *, vint *, vint *,
        void (*func)(int,int,real,mreal,vreal),
        void (*dfunc)(int,int,real,mreal,vreal,mmreal) );
extern void FREEPAR ( int, int, mreal *, mreal *, mreal *, vreal *, vint *, vint * );
extern void STATISTICS ( real *, int *, int *, int *, int *, int *, int *, int *, int *);
extern void STATUS ( int, char * );
```

5.3 Variáveis

O GSDAE utiliza uma variável global para armazenar todos os dados necessários para a integração utilizando as rotinas GSDAE e CSDAE. Esta variável chama-se "par" do tipo parmeter e é alocada na rotina ALLOCPAR e liberada na rotina FREEPAR. Todas as outras variáveis são locais, podendo serem estáticas ou não.

As variáveis de interface nas rotinas GSDAE, CSDAE, ALLOCPAR, e FREEPAR são:

- n : dimensão da EAD;
- o : ordem da EAD;
- h : passo de integração;
- hmin : passo minimo de integração;
- hmax : passo maximo de integração;
- cdmax : condição maxima para a derivada;
- s : c(s) = (x(s),y(s));
- x : c(s) = (x(s),y(s));
- y : c(s) = (x(s),y(s));
- send : ponto final de integração (rotina GSDAE);
- xend : ponto final de integração (rotina CSDAE);
- atolx : erro absoluto para x;

- atoly : erro absoluto para y;
- rtolx : erro realtivo para x;
- rtoly : erro realtivo para y;
- ftol : erro absoluto para a função;
- infoinput : informações para entrada de dados;
- infooutput : informações para saída de dados.

5.4 As Rotinas ALLOCPAR e FREEPAR

A alocação e liberação de memória no GSDAE é feita em duas rotinas: ALLOCPAR para alocar e FREEPAR para liberar. Estas rotinas são responsáveis pela alocação e liberação de todas as variáveis dinâmicas, sendo que as variáveis não utilizadas na interface, assim como a variável global "par", ficam invisíveis para o usuário.

Para o uso da rotina ALLOCPAR, as variáveis n e o devem ser definidas anteriormente.

```
int n, o;
vint infoinput, infooutput;
vreal ftol;
mmreal atoly, rtoly;
void func(int,int,real,mreal,vreal);
void dfunc(int,int,real,mreal,vreal,mmreal);

o = 2;
n = 5;


ALLOCPAR(n,q,&y,&atoly,&rtoly,&ftol,&infoinput,&infooutput,func,dfunc);

.
.
.
.

FREEPAR(n,q,&y,&atoly,&rtoly,&ftol,&infoinput,&infooutput);
```

5.5 As Rotinas func e dfunc Definidas pelo Usuário

As rotinas para avaliação da função F e sua derivada devem ser declaradas imdependentemente da derivada ser aproximada. Seus parâmetros são:

```
void func(int,int,real,mreal,vreal);
void func(int,int,real,mreal,vreal,mmreal);


void func ( int o, int n, real x, mreal y, vreal delta )
{
    delta[1] =
    .
    .
    delta[n] =

    return;
}


void dfunc ( int o, int n, real x, mreal y, vreal fx, mmreal fy )
{
    fx[1] =
    .
    .
    fx[n] =
    fy[0][1][1] =
    .
    .
    fy[0][n][n] =
    .
    .
    fy[o][1][1] =
    .
    .
    fy[o][n][n] =

    return;
}
```

5.6 As Rotinas GSDAE e CSDAE

A rotina GSDAE integra a EAD com condição inicial s,x,y até $s = send$ ou até uma singularidade transversal ou até ocorrer algum erro. A rotina CSDAE integra a EAD com condição inicial x,y até $x = xend$ ou até uma singularidade transversal ou até ocorrer algum erro. Na rotina CSDAE a condição inicial não pode ser uma singularidade transversal, pois como $x'(s) = 0$, não é possível saber a direção de integração na variável s para que a variável x chegue em $xend$.

Como este código é baseado na teoria apresentada na seção 2, ele é indicado para resolver EAD de índice 0 podendo conter singularidades (o posto de $DF_{y[o]}$ é no mínimo $n-1$) ou de índice 1 na forma semi-implícita podendo conter singularidades.

Dado um valor inicial, estas rotinas verificam se este ponto satisfaz a EAD, calculam o posto de $F_{y[o]}$ no ponto e em uma vizinhança, e de acordo com este posto (se $rank = rank(F_{y[o]}) < n$) são definidas as permutações das coordenadas de F e da variável y de modo que o posto se de nas primeiras linhas e colunas de $F_{y[o]}$. Caso o posto seja zero em uma vizinhança do ponto, a EAD é considerada de ordem $o-1$ e $F_{y[o-1]}$ com posto máximo em quase todo ponto da vizinhança, isto é, o ponto pode ser ainda uma singularidade transversal. Se a ordem for diminuida e o ponto não for um ponto regular ou uma singularidade transversal, um flag de erro fatal é retornado ao usuário. Este processo é utilizado quando a rotina GSDAE ou CSDAE é chamada pela primeira vez (ponto inicial) ou quando alguma instabilidade é detectada durante o processo de integração, pois pode ser devido a queda de posto.

Ambas as rotinas são funções que retornam um valor inteiro. O valor retornado esta entre -16 e 5 nesta versão. Este valor representa o estado da rotina no momento ou um erro detectado. O estado da rotina em condições de prosseguir são representados por inteiros entre 0 e 5, e os erros detectados por inteiros negativos entre -16 e -1.

Este valor representa:

- 0 : ponto regular (o valor send ou xend foi atingido);
- 1 : singularidade transversal;
- 2 : ponto regular e o posto de DFy[o] diminuiu;
- 3 : singularidade transversal e o posto de DFy[o] diminuiu;
- 4 : ponto regular e a ordem da EAD diminuiu;
- 5 : singularidade transversal e a ordem da EAD diminuiu;
- -1 : o parâmetro par não foi alocado;

- -2 : erro na entrada de dados;
- -3 : ponto inicial inadequado - não satisfaz a EAD com a tolerância desejada;
- -4 : posto maior que o informado;
- -5 : EAD com ordem zero e posto de DFy[0] zero;
- -6 : o posto de DFy[o] varia em uma vizinhaca do ponto;
- -7 : a ordem diminuiu e o posto de DFy[o] varia em uma vizinhaca do ponto;
- -8 : singularidade não tranversal;
- -9 : singularidade não transversal com posto de DFy[o] inferior;
- -10 : singularidade não transversal com ordem inferior;
- -11 : ponto inadequado - não satisfaz a EAD com a tolerância desejada;
- -12 : h < hmin;
- -13 : condição da jacobiana maior que a adimitida;
- -14 : a tolerancia desejada nao foi atingida na rotina corretora;
- -15 : houve falha na rotina e nehuma providencia foi tomada;
- -16 : ocorreu uma singularidade transversal na última chamada da rotina CSDAE e nehuma providencia foi tomada.

Se a rotina GSDAE for chamada com o código de erro -15, o programa é abortado. Se a rotina CSDAE for chamada com o código de erro -15 ou -16, o programa é abortado.

Parâmetros de entrada e saída:

- s : variável independente da parametrização da solução geral c - valor real;
- x : variável dependente x = x(s) que na EAD original e a variável independente - valor real;
- y : variável dependente y = y(s) que na EAD original e a variável dependente - matriz de numeros reais de dimensão o por n. y[0] representa a variável y, y[1] representa a primeira derivada de y, y[2] representa a segunda derivada de y, y[o] representa a derivada de ordem o de y;

Parâmetros de entrada:

- n : dimensão da EAD - inteiro maior que zero;
- o : ordem da EAD - inteiro maior ou igual a zero. Se o = 0 a equação é puramente algébrica;
- send : valor final de integração do parâmetro s - valor real (rotina GSDAE);
- xend : valor final de integração do parâmetro x - valor real (rotina CSDAE);
- h,hmin,hmax: passo atual, passo mínimo e passo máximo de integração - valores reais maiores ou iguais a zero.

  Se infoinput[1] = 0 (primeria chamada) e h = hmin = 0.0, são definidos por default hmin = 1.0e-16, h = 1.0e-15.

  Se infoinput[1] = 0 (primeria chamada), h = 0.0 e hmin != 0.0, h e definido por default h = 10.0*hmin.

  Se infoinput[1] = 1 (não for a primeria chamada) e 0.0 < hmin < par− >hmin, hmin é redefinido pelo valor de entrada.
- cdmax : condição máxima admitida para a matriz jacobiana - valor real maior ou igual a zero. Se cdmax = 0.0, cdmax é definido por default cdmax = 1.0e6;
- atolx,atoly: erro absoluto para c = (x,y) - atolx é um valor real maior ou igual a zero, atoly é uma matriz de dimensão o por n de valores reais maiores ou iguais a zero;
- rtolx,rtoly: erro relativo para c = (x,y) - rtolx é um valor real maior ou igual a zero, rtoly é uma matriz de dimensão o por n de valores reais maiores ou iguais a zero;
- ftol : erro absoluto para as coordenadas da função que define a EAD - vetor de dimensão n de valores reais. Se ftol[1] = 0.0, todo ponto é considerado satisfazendo a EAD no teste da rotina functionnorm, exceto na rotina settau que testa a condição inicial.

  Se infoinput[3] = 0 os valores de atolx, atoly, rtolx, rtoly e ftol são definidos por default rtolx = rtoly[i][j] = 0.0, atolx = atoly[i][j] = 1.0e-15 e ftol[i] = 1.0e-12.

  Se infoinput[3] = 1 os valores de atolx, rtolx, atoly, rtoly e ftol são definidos pelo usuário com escalares, isto é, atoly[i][j] = atolx, rtoly[i][j] = rtolx e ftol[i] = ftol[1].

  Se infoinput[3] = 2 todos os valores de atolx, rtolx, atoly, rtoly e ftol são definidos pelo usuário;
- infoinput : vetor de informações sobre a entrada de dados na rotina - vetor de dimensão 2n+10 de valores inteiros.

  infoinput[1]: infoinput[1] = 0 indica à rotina que é a primeira vez que ela é chamada. infoinput[1] = 1 indica à rotina que não é a primeira vez que ela é chamada. Quando infoinput[1] = 0, a rotina GSDAE ou CSDAE redefine infoinput[1] = 1;

infoinput[2]: infoinput[2] = 0 indica à rotina GSDAE ou CSDAE que a matriz jacobiana deve ser aproximada. infoinput[2] = 1 indica à rotina GSDAE ou CSDAE que deve ser utilizada a matriz jacobiana definida pelo usuario na rotina dfunc;

infoinput[3]: infoinput[3] = 0 indica à rotina que as tolerâncias atol, rtol e ftol devem ser definidas por default. infoinput[3] = 1 indica à rotina que as tolerâncias atol, rtol e ftol foram definidas pelo usuário como escalares. infoinput[3] = 2 indica à rotina que as tolerâncias atol, rtol e ftol foram definidas pelo usuário como vetoriais;

infoinput[4]: infoinput[4] = 0 indica à rotina que o posto de DFy[o] deve ser inicializado como sendo máximo, isto e, posto = n, e as permutações de coordendas da função e das variáveis y[0],..,y[o] devem ser inicializadas como sendo a identidade. infoinput[4] > 0 indica à rotina que o posto de DFy[o] é infoinput[4] e as permutações de coordenadas da função estão em infoinput[10+i] (i = 1..n) e as permutações das variáveis y[0],..,y[o] estão em infoinput[10+n+i] (i = 1..n);

infoinput[i]: i = 11..10+n armazena as permutações de coordenadas da função que define a EAD quando infoinput[4] > 0;

infoinput[i]: i = 11+n..10+2n armazena as permutações das variáveis y[0],..,y[o] quando infoinput[4] > 0;

infoinput[i]: i = 0,5..10 não são utilizadas nesta versão.

Parâmetros de saída:

- infooutput : informações sobre a saída de dados da rotina - vetor de dimensão 2n+10 de valores inteiros.

  infooutput[1]: informa o estado da integração na saída da rotina - contém o último valor retornado pela rotina.

  infooutput[2]: informa o posto de DFy[o].

  infooutput[3]: informa a ordem da EAD.

  infooutput[i] (i = 11..10+n): informa as permutações de coordenadas da função que define a EAD;

  infooutput[i] (i = 11+n..10+2n): informa as permutações das variáveis y[0],..,y[o];

  infooutput[i] (i = 0,5..10): não são utilizadas nesta versão.

```
status = GSDAE(n,o,h,hmin,hmax,cdmax,&s,send,&x,y,
                atolx,atoly,rtolx,rtoly,ftol,infoinput,&infooutput);
```

```
status = CSDAE(n,o,h,hmin,hmax,cdmax,&s,&x,xend,y,
               atolx,atoly,rtolx,rtoly,ftol,infoinput,infooutput);
```

Se status = 0, o valor *send* ou *xend* foi atingido e em x e y estará a solução. Se status > 0, ocorreu uma singularidade transversal, ou uma queda de posto de $F_{y[o]}$ ou a ordem o diminuiu antes de *xend* ou *send* ser atingido e em x e y estará o ponto. Se status $<$ 0, ocorreu um erro que GSDAE ou CSDAE não pode resolver e se a rotina STATUS for acionada voltará a menssagem do correspondente erro.

5.7 As Rotinas STATUS e STATISTICS

A rotina STATUS retorna o estado da integração no ponto retornado em um string a partir do valor status retornado por CSDAE ou GSDAE. Os parâmetros da rotina são:

```
int status;
char msg[250];

STATUS(status,msg);

printf("\n %s \n",msg);
```

O conteúdo da variável msg de acordo com o valor status é:

0 "Compleat execution : no error";

1 "Transversal singularity";

2 "Regular point and rank(DFy[o]) diminish";

3 "Transversal singularity and rank(DFy[o]) diminish";

4 "Regular point and order ¡ o";

5 "Transversal singularity and order ¡ o";

-1 "Incompleat execution : parameters not allocated";

-2 "Incompleat execution : error in data input";

-3 "Incompleat execution : initial point do not satisfies the DAE";

-4 "Incompleat execution : rank(DFy[0]) is greater than the actual";

-5 "Incompleat execution : order 0 and rank(DFy[0]) = 0";

-6 "Incompleat execution : rank(DFy[o]) is not constant in neighbourhood of the point";

-7 "Incompleat execution : order diminish and rank(DFy[o]) is not constant in neighbourhood of the point";

-8 "Incompleat execution : non transversal singularity";

-9 "Incompleat execution : non transversal singularity and rank(DFy[o]) diminish";

-10 "Incompleat execution : non transversal singularity and the order diminish";

-11 "Incompleat execution : point do not satisfies the DAE";

-12 "Incompleat execution : h ¡ hmin";

-13 "Incompleat execution : big condition matrix number";

-14 "Incompleat execution : the corrector could not converge";

-15 "Incompleat execution : the last step was terminated with a negative value and no appropriate action was taken";

-16 "Incompleat execution : the last step was terminated with a transversal singularity and no appropriate action was taken";

"Invalid value".

A rotina STATISTICS retorna dados sobre a integração. Seus parâmetros são:

int len, npas, nreject, nsuc, nfunc, njac, nqr, nstart, nfnew;

STATISTICS(&len,&npas,&nreject,&nsuc,&nfunc,&njac,&nqr,&nstart,&nfnew);

Nas variáveis retornam:

- len : comprimento de arco percorrido.
- npas : número de passos dados.
- nstart : número de inicializações.
- nreject : número de passos rejeitados.
- nsuc : número de passos com sucesso.
- nfnew : número de falhas na correção pelo método de Newton.
- nqr : número de decomposições QR.
- nfunc : número de avaliações da função F.
- njac : número de avaliações da derivada de F.

6 Exemplos

O primeiro exemplo é uma família de EAD a um parâmetro definida por:

$$\lambda(y')^2 + y^2 + x^2 = 1.$$

Esta família tem uma singularidade em $\lambda = 0$, isto é, para $\lambda \neq 0$ é uma EAD de primeira ordem e para $\lambda = 0$ é uma EAD de ordem zero (uma equação puramente algébrica). Mesmo para $\lambda \neq 0$, estas EAD contém singularidades transversais em $y' = 0$ e $y \neq \pm 1$.

O programa abaixo tem como entrada de dados o valor de λ e a condição inicial. O outros parâmetros são definidos no prórpio programa.

```
/*****************************************************/
/* Este programa utiliza o pacote GSDAE para integrar */
/* a familia de EAD a um parametro: */
/* lambda (y')**2 + y**2 + x**2 = 1 */
/*****************************************************/

/* incluindo as bibliotecas */
#include <stdio.h>
#include <math.h>

/* incluindo os tipos para o uso em GSDAE */
#include "types.h"

/* declarando as rotinas deste modulo */
void main ( void );
FILE *OPENF ( char *, char * );
void FESF ( int, int, real, mreal, vreal );
void DFESF ( int, int, real, mreal, vreal, mmreal );

/* declarando as rotinas do pacote GSDAE que serao */
/* utilizadas neste programa */
extern int GSDAE ( int, int, real, real, real, real, real *, real,
                   real *, mreal, real, mreal, real, mreal, vreal, vint, vint );
extern void ALLOCPAR ( int, int, mreal *, mreal *, mreal *, vreal *, vint *, vint *,
                   void (*F)(int,int,real,mreal,vreal),
                   void (*DF)(int,int,real,mreal,vreal,mmreal) );
extern void FREEPAR ( int, int, mreal *, mreal *, mreal *, vreal *, vint *, vint * );
extern void STATISTICS ( real *, int *, int *, int *, int *, int *, int *, int *, int *);
extern void STATUS ( int, char * );

/* declarando a variavel global */
real lambda;

/* programa principal */
void main ( void ) {

    /* declarando as variaveis locais utilizadas em GSDAE */
    int n;
```

```
int o;
real h;
real hmin;
real hmax;
real cdmax;
real s;
real send;
real x;
mreal y;
real atolx;
real rtolx;
mreal atoly;
mreal rtoly;
vreal ftol;
vint infoinput,infooutput;

/* delcarando as variaveis locais de controle */
int r;
int m;
real sep;
real ds;
int status;
char msg[250];
real len;
int npas;
int nreject;
int nsuc;
int nfnew;
int nstart;
int nfunc;
int njac;
int nqr;
real x0;
real y0;
real z0;

/* declarando o arquivo de saida de dados e seu nome */
FILE *file;
char name[30];
char nameout[50];

/* declarando as variaveis auxiliares */
int i,j,k,t,cont;

/* imprimindo a equacao */
printf("\n \n ");
printf("Equacao : lambda*(y')**2 + y**2 + x**2 = 1\n \n ");

printf("Enter output file name : ");
scanf("%s",name);
printf("\ n\ n");
```

```
/* definindo a dimensao e a ordem da EAD */
n = 1;
o = 1;

/* alocando os dados e */
/* inicializando a varavel global utilizada em GSDAE */
ALLOCPAR(n,o,&y,&atoly,&rtoly,&ftol,&infoinput,&infooutput,
            FESF,DFESF);

/* entrando com lambda */
printf("Enter lambda (-999.999 to stop) : ");
scanf("%lf",&lambda);
printf("\ n");

/* inicializando o contador de entrada de dados */
cont = 0;

/* laco para continuar a integracao */
while (lambda != -999.999) {

    /* entrando com a condicao inicial */
    printf("Enter x : ");
    scanf("%lf",&x);
    printf("Enter y : ");
    scanf("%lf",&(y[0][1]));
    printf("Enter y' : ");
    scanf("%lf",&(y[1][1]));
    printf("\ n\ n");

    /* armazenando o ponto inicial */
    x0 = x;
    y0 = y[0][1];
    z0 = y[1][1];

    /* integrando no sentido positivo do parametro s de */
    /* comprimento de arco */
    /* valor inicial para o parmetro s */
    s = 0.0;
    /* valor final para o parmetro s */
    send = 5.0;

    /* informando que e a primeira chamada da GSDAE */
    infoinput[1] = 0;

    /* informando que a jacobiana deve ser exata */
    /* dada em DFESF */
    infoinput[2] = 1;

    /* informando que as tolerancias serao dadas em */
    /* toda chamada da GSDAE */
    infoinput[3] = 2;

    /* definindo os espacametos */
```

```
/* espacamento inicial */
h = 1.0e-10;
/* espacamento minimo */
hmin = 1.0e-18;
/* espacamento maximo (ilimitado) */
hmax = 0.0;

/* definindo a condicao maxima para as jacobianas */
cdmax = 8.0e6;

/* definindo as tolerancias */
/* erros absoluto e relativo com relacao a x */
atolx = 1.0e-16;
rtolx = 1.0e-10;
/* erros absoluto e relativo com relacao a y */
atoly[0][1] = 1.0e-16;
rtoly[0][1] = 1.0e-10;
/* erros absoluto e relativo com relacao a y' */
atoly[1][1] = 1.0e-16;
rtoly[1][1] = 1.0e-10;
/* erro absoluto com relacao a F(x,y,y') */
ftol[1] = 1.0e-12;

/* imprimindo o ponto inicial */
printf("\ n\ nFirst point : \ n");
printf("s = %20.16lf \ n",s);
printf("x = %20.16lf \ n",x);
printf("y = %20.16lf \ n",y[0][1]);
printf("y' = %20.16lf \ n",y[1][1]);
printf("\ n");

/* abrindo o arquivo de saida para visualizar a solucao */
/* utilizando o codigo cvis */
sprintf(nameout,"%s.%d",name,2*cont);
file = (FILE *) OPENF(nameout,"w");
fprintf(file,"4\ n");

/* escrevendo o ponto inicial no arquivo de saida */
fprintf(file,"1 ");
fprintf(file,"%20.16lf ",s);
fprintf(file,"%20.16lf ",x);
fprintf(file,"%20.16lf ",y[0][1]);
fprintf(file,"%20.16lf ",y[1][1]);
fprintf(file,"\ n");

/* definindo 1000 pontos entre o ponto inicial e o */
/* ponto final */
m = 1000;
ds = (send-s)/(real)m;
sep = s;

/* definindo o ponto inicial como ponto regular */
status = 0;
```

```
/* integrando ate o ponto final ou uma falha */
for (t = 1; (t <= m) && (status >= 0); t++) {

    /* incrementando o ponto final parcial */
    sep += ds;

    /* laco de integracao ate o ponto final */
    do {

        /* integrando a EAD */
        status = GSDAE(n,o,h,hmin,hmax,cdmax,&s,sep,&x,y,
                       atolx,atoly,rtolx,rtoly,ftol,infoinput,infooutput);

        /* verificando o tipo de saida */
        if (status == 0) {

            /* ponto de saida regular */

            /* escrevendo o ponto de saida no arquivo de saida */
            fprintf(file,"1 ");
            fprintf(file,"%20.16lf ",s);
            fprintf(file,"%20.16lf ",x);
            fprintf(file,"%20.16lf ",y[0][1]);
            fprintf(file,"%20.16lf ",y[1][1]);
            fprintf(file,"\ n");

        } else if (status > 0) {

            /* ponto de saida e uma singularidade transversal */

            /* escrevendo o ponto de saida no arquivo de saida */
            fprintf(file,"2 ");
            fprintf(file,"%20.16lf ",s);
            fprintf(file,"%20.16lf ",x);
            fprintf(file,"%20.16lf ",y[0][1]);
            fprintf(file,"%20.16lf ",y[1][1]);
            fprintf(file,"\ n");

            /* imprimindo a menssagem sobre o ponto de saida */
            STATUS(status,msg);
            printf("%s\ n",msg);

            /* imprimindo o ponto de saida */
            printf("s = %20.16lf \ n",s);
            printf("x = %20.16lf \ n",x);
            printf("y = %20.16lf \ n",y[0][1]);
            printf("y' = %20.16lf \ n",y[1][1]);
            printf("\ n");

        }

    } while (status > 0);

}
```

```
/* imprimindo o ponto final */
printf("\ n\ nLast point : \ n");
printf("s = %20.16lf \ n",s);
printf("x = %20.16lf \ n",x);
printf("y = %20.16lf \ n",y[0][1]);
printf("y' = %20.16lf \ n",y[1][1]);
printf("\ n");

/* obtendo a menssagem sobre o ponto final */
STATUS(status,msg);

/* obtendo a estatistica sobre a integracao */
STATISTICS(&len,&npas,&nreject,&nsuc,&nfunc,&njac,&nqr,&nstart,&nfnew);

/* verificando se houve erro na integracao */
if (status < 0) {

    /* houve erro na integracao */
    printf("\ n\ nError message\ n");

    /* escrevendo o ponto final no arquivo de saida */
    fprintf(file,"3 ");
    fprintf(file,"%20.16lf ",s);
    fprintf(file,"%20.16lf ",x);
    fprintf(file,"%20.16lf ",y[0][1]);
    fprintf(file,"%20.16lf ",y[1][1]);
    fprintf(file,"\ n");

}

/* imprimindo a menssagem de erro */
printf("%s\ n",msg);

/* imprimindo a estatistica da integracao */
printf("\ n\ nStatistics\ n");
printf("Arc-lenght : %lf\ n",len);
printf("Number of Steps : %d\ n",npas);
printf("Number of Rejected Steps : %d\ n",nreject);
printf("Number of Success Steps : %d\ n",nsuc);
printf("Number of Starts : %d\ n",nstart);
printf("Number of Failed Newton Steps : %d\ n",nfnew);
printf("Number of Evaluations of the Function : %d\ n",nfunc);
printf("Number of Evaluations of the Jacobian : %d\ n",njac);
printf("Number of QR Decompositions : %d\ n",nqr);

/* fechando o arquivo de saida */
fclose(file);

/* integrando no sentido negativo do parametro s de */
/* comprimento de arco */
/* valor inicial para o parmetro s */
s = 0.0;
/* valor final para o parmetro s */
```

```
send = -5.0;

/* recupernado a condicao inicial */
x = x0;
y[0][1] = y0;
y[1][1] = z0;

/* informando que e a primeira chamada da GSDAE */
infoinput[1] = 0;

/* informando que a jacobiana deve ser exata */
/* dada em DFESF */
infoinput[2] = 1;

/* informando que as tolerancias serao dadas em */
/* toda chamada da GSDAE */
infoinput[3] = 2;

/* definindo os espacametos */
/* espacamento inicial */
h = 1.0e-10;
/* espacamento minimo */
hmin = 1.0e-18;
/* espacamento maximo (ilimitado) */
hmax = 0.0;

/* definindo a condicao maxima para as jacobianas */
cdmax = 8.0e6;

/* definindo as tolerancias */
/* erros absoluto e relativo com relacao a x */
atolx = 1.0e-16;
rtolx = 1.0e-10;
/* erros absoluto e relativo com relacao a y */
atoly[0][1] = 1.0e-16;
rtoly[0][1] = 1.0e-10;
/* erros absoluto e relativo com relacao a y' */
atoly[1][1] = 1.0e-16;
rtoly[1][1] = 1.0e-10;
/* erro absoluto com relacao a F(x,y,y') */
ftol[1] = 1.0e-12;

/* imprimindo o ponto inicial */
printf("\ n\ nFirst point : \ n");
printf("s = %20.16lf \ n",s);
printf("x = %20.16lf \ n",x);
printf("y = %20.16lf \ n",y[0][1]);
printf("y' = %20.16lf \ n",y[1][1]);
printf("\ n");

/* abrindo o arquivo de saida para visualizar a solucao */
/* utilizando o codigo cvis */
sprintf(nameout,"%s.%d",name,2*cont+1);
```

```
file = (FILE *) OPENF(nameout,"w");
fprintf(file,"4\ n");

/* escrevendo o ponto inicial no arquivo de saida */
fprintf(file,"1 ");
fprintf(file,"%20.16lf ",s);
fprintf(file,"%20.16lf ",x);
fprintf(file,"%20.16lf ",y[0][1]);
fprintf(file,"%20.16lf ",y[1][1]);
fprintf(file,"\ n");

/* definindo 1000 pontos entre o ponto inicial e o */
/* ponto final */
m = 1000;
ds = (send-s)/(real)m;
sep = s;

/* definindo o ponto inicial como ponto regular */
status = 0;

/* integrando ate o ponto final ou uma falha */
for (t = 1; (t <= m) && (status >= 0); t++) {

    /* incrementando o ponto final parcial */
    sep += ds;

    /* laco de integracao ate o ponto final */
    do {

        /* integrando a EAD */
        status = GSDAE(n,o,h,hmin,hmax,cdmax,&s,sep,&x,y,
                       atolx,atoly,rtolx,rtoly,ftol,infoinput,infooutput);

        /* verificando o tipo de saida */
        if (status == 0) {

            /* ponto de saida regular */

            /* escrevendo o ponto de saida no arquivo de saida */
            fprintf(file,"1 ");
            fprintf(file,"%20.16lf ",s);
            fprintf(file,"%20.16lf ",x);
            fprintf(file,"%20.16lf ",y[0][1]);
            fprintf(file,"%20.16lf ",y[1][1]);
            fprintf(file,"\ n");

        } else if (status > 0) {

            /* ponto de saida e uma singularidade transversal */

            /* escrevendo o ponto de saida no arquivo de saida */
            fprintf(file,"2 ");
            fprintf(file,"%20.16lf ",s);
```

```
            fprintf(file,"%20.16lf ",x);
            fprintf(file,"%20.16lf ",y[0][1]);
            fprintf(file,"%20.16lf ",y[1][1]);
            fprintf(file,"\ n");

            /* imprimindo a menssagem sobre o ponto de saida */
            STATUS(status,msg);
            printf("%s\ n",msg);

            /* imprimindo o ponto de saida */
            printf("s = %20.16lf \ n",s);
            printf("x = %20.16lf \ n",x);
            printf("y = %20.16lf \ n",y[0][1]);
            printf("y' = %20.16lf \ n",y[1][1]);
            printf("\ n");

        }

    } while (status > 0);

}

/* imprimindo o ponto final */
printf("\ n\ nLast point : \ n");
printf("s = %20.16lf \ n",s);
printf("x = %20.16lf \ n",x);
printf("y = %20.16lf \ n",y[0][1]);
printf("y' = %20.16lf \ n",y[1][1]);
printf("\ n");

/* obtendo a menssagem sobre o ponto final */
STATUS(status,msg);

/* obtendo a estatistica sobre a integracao */
STATISTICS(&len,&npas,&nreject,&nsuc,&nfunc,&njac,&nqr,&nstart,&nfnew);

/* verificando se houve erro na integracao */
if (status < 0) {

    /* houve erro na integracao */
    printf("\ n\ nError message\ n");

    /* escrevendo o ponto final no arquivo de saida */
    fprintf(file,"3 ");
    fprintf(file,"%20.16lf ",s);
    fprintf(file,"%20.16lf ",x);
    fprintf(file,"%20.16lf ",y[0][1]);
    fprintf(file,"%20.16lf ",y[1][1]);
    fprintf(file,"\ n");

}

/* imprimindo a menssagem de erro */
```

```
        printf("%s\ n",msg);

        /* imprimindo a estatistica da integracao */
        printf("\ n\ nStatistics\ n");
        printf("Arc-lenght : %lf\ n",len);
        printf("Number of Steps : %d\ n",npas);
        printf("Number of Rejected Steps : %d\ n",nreject);
        printf("Number of Success Steps : %d\ n",nsuc);
        printf("Number of Starts : %d\ n",nstart);
        printf("Number of Failed Newton Steps : %d\ n",nfnew);
        printf("Number of Evaluations of the Function : %d\ n",nfunc);
        printf("Number of Evaluations of the Jacobian : %d\ n",njac);
        printf("Number of QR Decompositions : %d\ n",nqr);

        /* fechando o arquivo de saida */
        fclose(file);

        /* incrementando o contador de entradas */
        cont++;

        /* entrando com lambda */
        printf("\ n\ n");
        printf("Enter lambda (-999.999 to stop) : ");
        scanf("%lf",&lambda);
        printf("\ n");

    }

    /* liberando todos os dados da memoria */
    FREEPAR(n,o,&y,&atoly,&rtoly,&ftol,&infoinput,&infooutput);

    printf("\ n\ n\ n");

    return;
}

/*******************************/
/* rotina para abrir um arquivo */
/*******************************/

FILE *OPENF( name,type )
char *name; /* nome do arquivo */
char *type; /* tipo de abertura : r para ler, w para escrever, .. */
{
    FILE *f;

    if ((f = fopen(name,type)) == NULL) {
        printf("\ n");
        printf(" ERRO : arquivo %s nao aberto\ n",name);
        printf("\ n");
        exit (1);
    }
```

```
    return (f);
}

/************************************/
/* rotina que define a EAD */
/* lambda (y')**2 + y**2 + x**2 = 1 */
/************************************/

void FESF( o, n, x, y, delta )
int o;
int n;
real x;
mreal y;
vreal delta ;
{       /* primeira coordenada */
    delta[1] = x*x+y[0][1]*y[0][1]+lambda*y[1][1]*y[1][1]-1.0;

    return;
}

/*****************************************/
/* rotina que define a jacobiana da EAD */
/* lambda (y')**2 + y**2 + x**2 = 1 */
/*****************************************/

void DFESF( o, n, x, y, DFx, DFy )
int o;
int n;
real x;
mreal y;
vreal DFx;
mmreal DFy;
{
    /* derivada com relacao a x */
    DFx[1] = 2.0*x;

    /* derivada com relacao a y */
    DFy[0][1][1] = 2.0*y[0][1];

    /* derivada com relacao a y' */
    DFy[1][1][1] = 2.0*lambda*y[1][1];

    return;
}
```

Execução do programa:

```
Equation : lambda*(y')**2 + y**2 + x**2 = 1

Enter output file name : esf

Enter lambda (-999.999 to stop) : -1
```

Enter x : 1
Enter y : 0
Enter y' : 0

First point :
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Last point :
s = 4.9999999999999156
x = 2.5719253971811669
y = 2.7775780117848305
y' = 3.6509916269710274

Compleat execution : no error

Statistics
Arc-lenght : 5.003215
Number of Steps : 498
Number of Rejected Steps : 7
Number of Success Steps : 491
Number of Starts : 1
Number of Failed Newton Step : 11
Number of Evaluations of the Function : 1663
Number of Evaluations of the Jacobian : 69
Number of QR Decompositions : 64

First point :
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Last point :
s = -4.9999999999999156
x = 2.5719253971811669
y = -2.7775780117848305
y' = -3.6509916269710274

Compleat execution : no error

Statistics
Arc-lenght : -5.003215
Number of Steps : 498
Number of Rejected Steps : 7
Number of Success Steps : 491
Number of Starts : 1
Number of Failed Newton Steps : 11
Number of Evaluations of the Function : 1663
Number of Evaluations of the Jacobian : 69
Number of QR Decompositions : 64

Enter lambda (-999.999 to stop) : 0

Enter x : 1
Enter y : 0
Enter y' : 0

First point :
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity and order < o
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity
s = 3.1415926535914047
x = -0.9999999999997845
y = 0.0000000000077140
y' = 0.0000000000000000

Last point :
s = 4.9999999999999156
x = 0.2836621854651625
y = -0.9589242746620048
y' = 0.0000000000000000

Compleat execution : no error

Statistics Arc-lenght : 5.013623
Number of Steps : 339
Number of Rejected Steps : 1
Number of Success Steps : 338
Number of Starts : 1
Number of Failed Newton Steps : 14
Number of Evaluations of the Function : 1257
Number of Evaluations of the Jacobian : 78

Number of QR Decompositions : 70

First point :
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity and order < o
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity
s = -3.1415926535914047
x = -0.9999999999997845
y = -0.0000000000077140
y' = 0.0000000000000000

Last point :
s = -4.9999999999999156
x = 0.2836621854651625
y = 0.9589242746620048
y' = 0.0000000000000000

Compleat execution : no error

Statistics
Arc-lenght : -5.013623
Number of Steps : 339
Number of Rejected Steps : 1
Number of Success Steps : 338
Number of Starts : 1
Number of Failed Newton Steps : 14
Number of Evaluations of the Function : 1257
Number of Evaluations of the Jacobian : 78
Number of QR Decompositions : 70

Enter lambda (-999.999 to stop) : 1

Enter x : 1
Enter y : 0
Enter y' : 0

First point :
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity
s = 0.0000000000000000

x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity
s = 2.6430622099737011
x = -0.4265700936557158
y = -0.9044545069811715
y' = 0.0000000000042041

Transversal singularity
s = 3.4201058204710471
x = 0.0753251119040765
y = -0.9971590281979291
y' = -0.0000000000000141

Transversal singularity
s = 3.5530771769329696
x = -0.0123093017343699
y = -0.9999242376754411
y' = 0.0000000000125090

Last point :
s = 3.5549999999999464
x = -0.0101955293805785
y = -0.9999326925337505
y' = -0.0055372902085959

Error message Incompleat execution : h < hmin

Statistics
Arc-lenght : 3.559197
Number of Steps : 1588
Number of Rejected Steps : 39
Number of Success Steps : 1549
Number of Starts : 2
Number of Failed Newton Steps : 59
Number of Evaluations of the Function : 5205
Number of Evaluations of the Jacobian : 214
Number of QR Decompositions : 208

First point :
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity
s = 0.0000000000000000
x = 1.0000000000000000
y = 0.0000000000000000
y' = 0.0000000000000000

Transversal singularity
s = -2.6430622099737011
x = -0.4265700936557158
y = 0.9044545069811715
y' = -0.0000000000042041

Transversal singularity
s = -3.4201058204710471
x = 0.0753251119040765
y = 0.9971590281979291
y' = 0.0000000000000141

Transversal singularity
s = -3.5530771769329696
x = -0.0123093017343699
y = 0.9999242376754411
y' = -0.0000000000125090

Last point :
s = -3.5549999999999464
x = -0.0101955293805785
y = 0.9999326925337505
y' = 0.0055372902085959

Error message
Incompleat execution : h < hmin

Statistics
Arc-lenght : -3.559197
Number of Steps : 1588
Number of Rejected Steps : 39
Number of Success Steps : 1549
Number of Starts : 2
Number of Failed Newton Steps : 59
Number of Evaluations of the Function : 5205
Number of Evaluations of the Jacobian : 214
Number of QR Decompositions : 208

Enter lambda (-999.999 to stop) : -999.999

Abaixo tem-se a saida gráfica deste programa. A figura 1 exibe o plano (s, x) na qual fica facil de perceber as singularidades tranversais, pontos estes em que $x' = 0$. A figura 2 exibe o plano (x, y) e a figura 3 o plano (x, y').

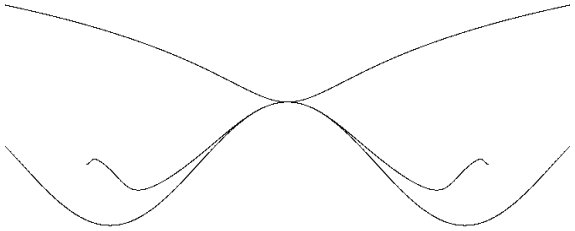

Figura 1 : plano (s, x)

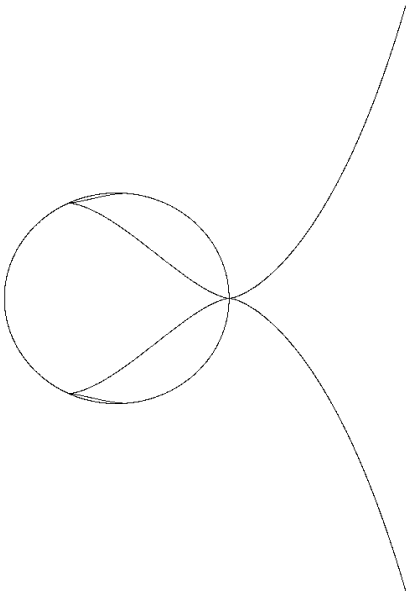

Figura 2 : plano (x, y)

Figura 3 : plano (x, y')

Para o mesmo exemplo com $h = 1.0e-8$, $hmin = 1.0e-14$, $hmax = 0.0$, $cdmax = 8.0e6$, $atolx = 1.0e-15$, $rtolx = 1.0e-8$, $atoly[0][1] = 1.0e-15$, $rtoly[0][1] = 1.0e-8$, $atoly[1][1] = 1.0e-15$, $rtoly[1][1] = 1.0e-8$ e $ftol[1] = 0.0$, a saída é:

Equation : lambda*(y')**2 + y**2 + x**2 = 1

Enter output file name : esf

Enter lambda (-999.999 to stop) : 1.0

Enter x : 0.0
Enter y : 0.0
Enter y' : 1.0

First point :
s = 0.0000000000000000
x = 0.0000000000000000
y = 0.0000000000000000
y' = 1.0000000000000000

Transversal singularity
s = 1.5756294078819535
x = -0.7975004284944297
y = -0.6033183790981006
y' = -0.0000000024400595

Transversal singularity
s = 3.1957161495502517
x = 0.1838599507253551
y = -0.9829524497753855
y' = 0.0000000064135492

Transversal singularity
s = 3.5218297216097807
x = -0.0304003073262786
y = -0.9995378038446170
y' = -0.0000000007265379

Transversal singularity
s = 3.5754527920707826
x = 0.0049581565681774
y = -0.9999877082661793
y' = 0.0000000000149865

Transversal singularity
s = 3.5841971684120622
x = -0.0008083540395983
y = -0.9999996732818199
y' = -0.0000000004656834

Last point :
s = 3.5849999999999347
x = -0.0000091609539873
y = -0.9999999686496943
y' = -0.0002502332658898

Error message

Incompleat execution : h < hmin

Statistics
Arc-lenght : 3.585329
Number of Steps : 2069
Number of Rejected Steps : 116
Number of Success Steps : 1953
Number of Starts : 7
Number of Failed Newton Steps : 103
Number of Evaluations of the Function : 3958
Number of Evaluations of the Jacobian : 419
Number of QR Decompositions : 412

First point :
s = 0.0000000000000000
x = 0.0000000000000000
y = 0.0000000000000000
y' = 1.0000000000000000

Transversal singularity
s = -1.5756294078819535
x = 0.7975004284944297
y = 0.6033183790981006
y' = -0.0000000024400595

Transversal singularity
s = -3.1957161495502517
x = -0.1838599507253551
y = 0.9829524497753855
y' = 0.0000000064135492

Transversal singularity
s = -3.5218297216097807
x = 0.0304003073262786
y = 0.9995378038446170
y' = -0.0000000007265379

Transversal singularity
s = -3.5754527920707826
x = -0.0049581565681774
y = 0.9999877082661793
y' = 0.0000000000149865

Transversal singularity
s = -3.5841971684120622
x = 0.0008083540395983
y = 0.9999996732818199
y' = -0.0000000004656834

Last point :
s = -3.5849999999999347
x = 0.0000091609539873
y = 0.9999999686496943

y’ = -0.0002502332658898

Error message
Incompleat execution : h < hmin

Statistics
Arc-lenght : -3.585329
Number of Steps : 2069
Number of Rejected Steps : 116
Number of Success Steps : 1953
Number of Starts : 7
Number of Failed Newton Steps : 103
Number of Evaluations of the Function : 3958
Number of Evaluations of the Jacobian : 419
Number of QR Decompositions : 412

Enter lambda (-999.999 to stop) : 1.0

Enter x : 0.0
Enter y : 0.0
Enter y’ : -1.0

First point :
s = 0.0000000000000000
x = 0.0000000000000000
y = 0.0000000000000000
y’ = -1.0000000000000000

Transversal singularity
s = 1.5756294078819535
x = 0.7975004284944297
y = -0.6033183790981006
y’ = 0.0000000024400595

Transversal singularity
s = 3.1957161495502517
x = -0.1838599507253551
y = -0.9829524497753855
y’ = -0.0000000064135492

Transversal singularity
s = 3.5218297216097807
x = 0.0304003073262786
y = -0.9995378038446170
y’ = 0.0000000007265379

Transversal singularity
s = 3.5754527920707826
x = -0.0049581565681774
y = -0.9999877082661793
y’ = -0.0000000000149865

Transversal singularity

s = 3.5841971684120622
x = 0.0008083540395983
y = -0.9999996732818199
y' = 0.0000000004656834

Last point :
s = 3.5849999999999347
x = 0.0000091609539873
y = -0.9999999686496943
y' = 0.0002502332658898

Error message
Incompleat execution : h < hmin

Statistics
Arc-lenght : 3.585329
Number of Steps : 2069
Number of Rejected Steps : 116
Number of Success Steps : 1953
Number of Starts : 7
Number of Failed Newton Steps : 103
Number of Evaluations of the Function : 3958
Number of Evaluations of the Jacobian : 419
Number of QR Decompositions : 412

First point :
s = 0.0000000000000000
x = 0.0000000000000000
y = 0.0000000000000000
y' = -1.0000000000000000

Transversal singularity
s = -1.5756294078819535
x = -0.7975004284944297
y = 0.6033183790981006
y' = 0.0000000024400595

Transversal singularity
s = -3.1957161495502517
x = 0.1838599507253551
y = 0.9829524497753855
y' = -0.0000000064135492

Transversal singularity
s = -3.5218297216097807
x = -0.0304003073262786
y = 0.9995378038446170
y' = 0.0000000007265379

Transversal singularity
s = -3.5754527920707826
x = 0.0049581565681774
y = 0.9999877082661793

y' = -0.0000000000149865

Transversal singularity
s = -3.5841971684120622
x = -0.0008083540395983
y = 0.9999996732818199
y' = 0.0000000004656834

Last point :
s = -3.5849999999999347
x = -0.0000091609539873
y = 0.9999999686496943
y' = 0.0002502332658898

Error message
Incompleat execution : h < hmin

Statistics
Arc-lenght : -3.585329
Number of Steps : 2069
Number of Rejected Steps : 116
Number of Success Steps : 1953
Number of Starts : 7
Number of Failed Newton Steps : 103
Number of Evaluations of the Function : 3958
Number of Evaluations of the Jacobian : 419
Number of QR Decompositions : 412

Enter lambda (-999.999 to stop) : -999.999

A saida gráfica deste programa é apresentada abaixo. A figura 4 exibe o plano (s, x) A figura 5, 6, 7 e 8 exibe o plano (x, y') focando as singularidades não transversais ($x = y' = 0$, $y = \pm 1$).

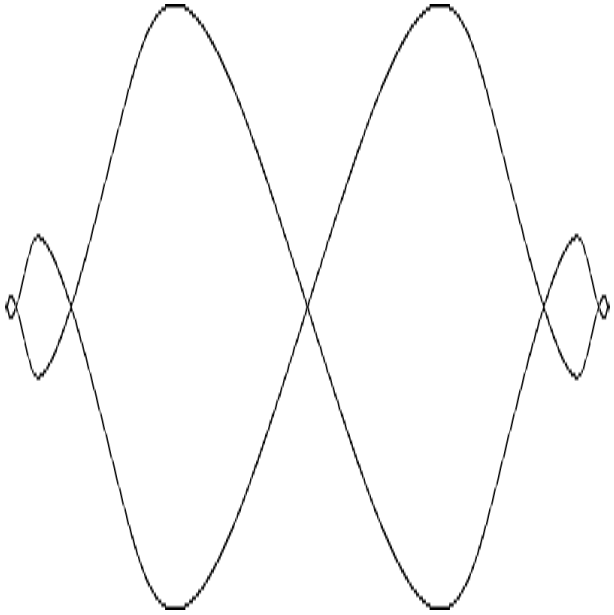

Figura 4 : plano (s, x)

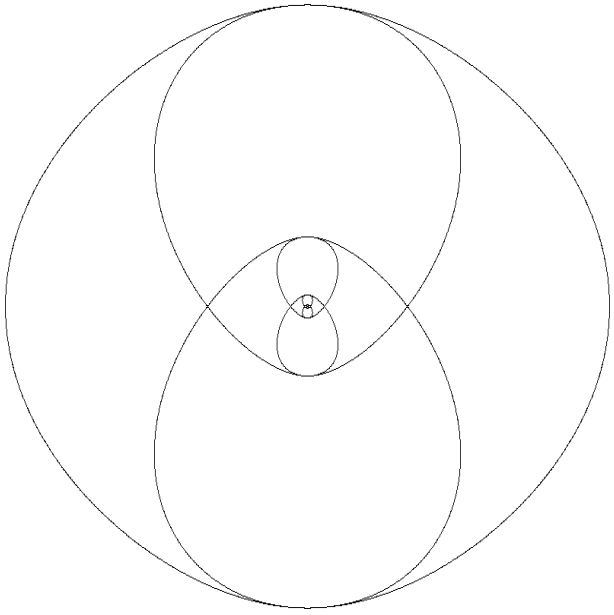

Figura 5 : plano (x, y')

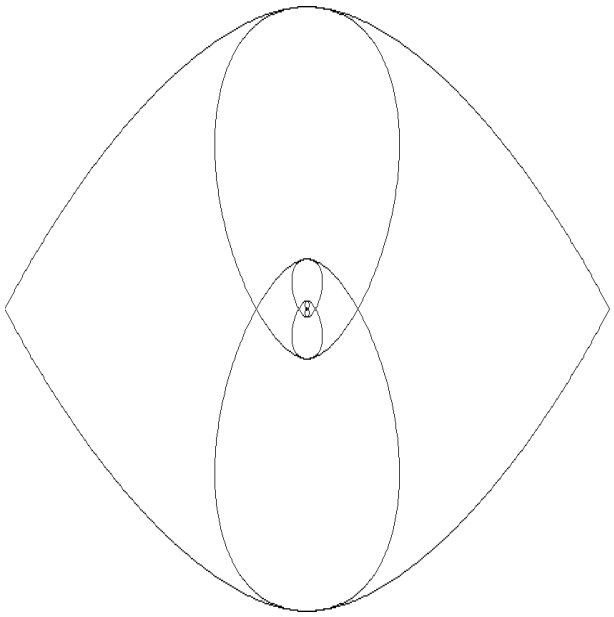

Figura 6 : plano (x, y') ampliado

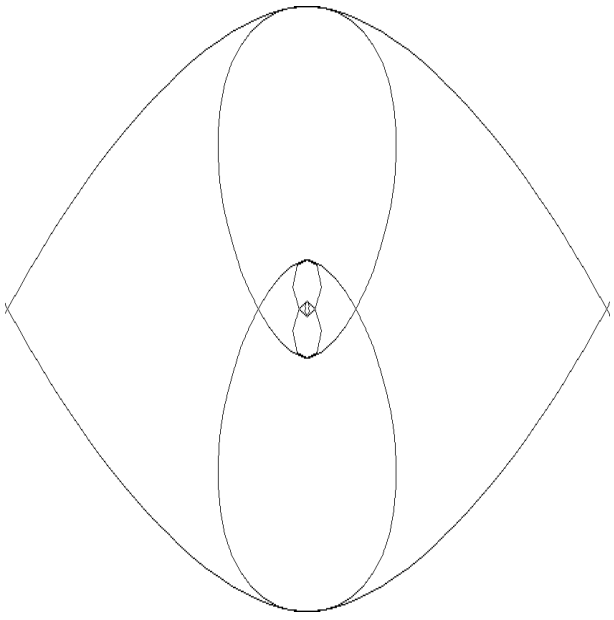

Figura 7 : plano (x, y') duas vezes ampliado

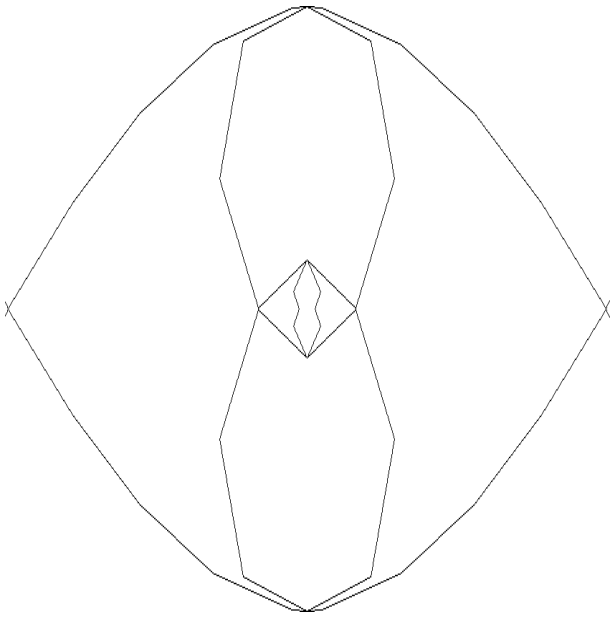

Figura 8 : plano (x, y') três vezes ampliado

O segundo exemplo é a clássica EDO

$$y' = y.$$

Para este exemplo, a rotina CSDAE é mais adequada, pois é interessante integrar até um valor de x. Neste exemplo também será obtido o erro relativo em cada ponto.

```
/*****************************************************/
/* Este programa utiliza o pacote GSDAE para integrar */
/* y' - y = 0 */
/*****************************************************/

/* incluindo as bibliotecas */
#include <stdio.h>
#include <math.h>

/* incluindo os tipos para o uso em GSDAE */
#include "types.h"

/* declarando as rotinas deste modulo */
void main ( void );
FILE *OPENF ( char *, char * );
void FEXP ( int, int, real, mreal, vreal );
void DFEXP ( int, int, real, mreal, vreal, mmreal );

/* declarando as rotinas do pacote GSDAE que serao */
/* utilizadas neste programa */
extern int CSDAE ( int, int, real, real, real, real, real *, real *,
real, mreal, real, mreal, real, mreal, vreal,
```

```
vint, vint );
extern void ALLOCPAR ( int, int, mreal *, mreal *, mreal *,
vreal *, vint *, vint *,
void (*F)(int,int,real,mreal,vreal),
void (*DF)(int,int,real,mreal,vreal,mmreal) );
extern void FREEPAR ( int, int, mreal *, mreal *, mreal *, vreal *,
vint *, vint * );
extern void STATISTICS ( real *, int *, int *, int *, int *, int *,
int *, int *, int *);
extern void STATUS ( int, char * );

/* programa principal */
void main ( void )
{

    /* declarando as variaveis locais utilizadas em GSDAE */
    int n;
    int o;
    real h;
    real hmin;
    real hmax;
    real cdmax;
    real s;
    real xend;
    real x;
    mreal y;
    real atolx;
    real rtolx;
    mreal atoly;
    mreal rtoly;
    vreal ftol;
    vint infoinput;
    vint infooutput;

    /* delcarando as variaveis locais de controle */
    int r;
    int m;
    real xep;
    real dx;
    int status;
    char msg[250];
    real len;
    int npas;
    int nreject;
    int nsuc;
    int nfnew;
    int nstart;
    int nfunc;
    int njac;
    int nqr;
    real error;
```

```
/* declarando as variaveis auxiliares */
int i,j,k,t;

/* imprimindo a equacao */
printf("\n\n");
printf("Equation : y' = y\n\n");

/* definindo a dimensao e a ordem da EAD */
n = 1;
o = 1;

/* alocando os dados e */
/* inicializando a varavel global utilizada em GSDAE */
ALLOCPAR(n,o,&y,&atoly,&rtoly,&ftol,&infoinput,&infooutput, FEXP,DFEXP);

/* entrando com a condicao inicial */
printf("Enter x : ");
scanf("%lf",&x);
printf("Enter y : ");
scanf("%lf",&(y[0][1]));
printf("Enter y' : ");
scanf("%lf",&(y[1][1]));
printf("\n\n");
printf("Enter xend : ");
scanf("%lf",&xend);

/* integrando no sentido positivo do parametro s de */
/* comprimento de arco */
/* valor inicial para o parmetro s */
s = 0.0;

/* informando que e a primeira chamada da GSDAE */
infoinput[1] = 0;

/* informando que a jacobiana deve ser exata */
/* dada em DFEXP */
infoinput[2] = 1;

/* informando que as tolerancias serao dadas em */
/* toda chamada da GSDAE */
infoinput[3] = 2;

/* definindo os espacametos */
/* espacamento inicial */
h = 1.0e-10;
/* espacamento minimo */
hmin = 1.0e-15;
/* espacamento maximo (ilimitado) */
hmax = 0.0;

/* definindo a condicao maxima para as jacobianas */
cdmax = 1.0e50;
```

```
/* definindo as tolerancias */
/* erros absoluto e relativo com relacao a x */
atolx = 1.0e-15;
rtolx = 1.0e-10;
/* erros absoluto e relativo com relacao a y */
atoly[0][1] = 1.0e-15;
rtoly[0][1] = 1.0e-10;
/* erros absoluto e relativo com relacao a y' */
atoly[1][1] = 1.0e-15;
rtoly[1][1] = 1.0e-10;
/* erro absoluto com relacao a F(x,y,y') */
ftol[1] = 0.0;

/* calculando o erro */
error = fabs(y[0][1]-exp(x))/exp(x);

/* imprimindo o ponto inicial */
printf("\n\nFirst point : \n");
printf("s = %20.16lf \n",s);
printf("x = %20.16lf \n",x);
printf("y = %20.16lf \n",y[0][1]);
printf("y' = %20.16lf \n",y[1][1]);
printf("relative error = %20.16lf \n",error);
printf("\n");

/* definindo 10 pontos entre o ponto inicial e o */
/* ponto final */
m = 10;
dx = (xend-x)/(real)m;
xep = x;

/* definindo o ponto inicial como ponto regular */
status = 0;

/* integrando ate o ponto final ou uma falha */
for (t = 1; (t ≤ m) && (status == 0); t++) {

    /* incrementando o ponto final parcial */
    xep += dx;

    /* integrando a EAD */
    status = CSDAE(n,o,h,hmin,hmax,cdmax,&s,&x,xep,y,
                   atolx,atoly,rtolx,rtoly,ftol,infoinput, infooutput);

    /* verificando o tipo de saida */
    if (status == 0) {

        /* ponto de saida regular */

        /* calculando o erro */
        error = fabs(y[0][1]-exp(x))/exp(x);

        /* imprimindo o ponto de saida */
```

```
            printf("s = %20.16lf \n",s);
            printf("x = %20.16lf \n",x);
            printf("y = %20.16lf \n",y[0][1]);
            printf("y' = %20.16lf \n",y[1][1]);
            printf("relative error = %20.16lf \n",error);
            printf("\n");

        }

}

/* calculando o erro */
error = fabs(y[0][1]-exp(x))/exp(x);

/* imprimindo o ponto final */
printf("\n\nLast point : \n");
printf("s = %20.16lf \n",s);
printf("x = %20.16lf \n",x);
printf("y = %20.16lf \n",y[0][1]);
printf("y' = %20.16lf \n",y[1][1]);
printf("relative error = %20.16lf \n",error);
printf("\n");

/* obtendo a menssagem sobre o ponto final */
STATUS(status,msg);

/* obtendo a estatistica sobre a integracao */
STATISTICS(&len,&npas,&nreject,&nsuc,&nfunc,&njac,&nqr,&nstart,&nfnew);

/* verificando se houve erro na integracao */
if (status < 0) {

/* houve erro na integracao */
     printf("\n\nError message\n");

}

/* imprimindo a menssagem de erro */
printf("%s\n",msg);

/* imprimindo a estatistica da integracao */
printf("\n\nStatistics\n");
printf("Arc-lenght : %lf\n",len);
printf("Number of Steps : %d\n",npas);
printf("Number of Rejected Steps : %d\n",nreject);
printf("Number of Success Steps : %d\n",nsuc);
printf("Number of Starts : %d\n",nstart);
printf("Number of Failed Newton Steps : %d\n",nfnew);
printf("Number of Evaluations of the Function : %d\n",nfunc);
printf("Number of Evaluations of the Jacobian : %d\n",njac);
printf("Number of QR Decompositions : %d\n",nqr);

/* liberando todos os dados da memoria */
```

```
    FREEPAR(n,o,&y,&atoly,&rtoly,&ftol,&infoinput,&infooutput);

    printf("\n\n\n");

    return;

}

/*******************************/
/* rotina para abrir um arquivo */
/*******************************/

FILE *OPENF( name,type )
char *name; /* nome do arquivo */
char *type; /* tipo de abertura : r para ler, w para escrever, .. */
{

    FILE *f;

    if ((f = fopen(name,type)) == NULL) {

        printf("\n");
        printf(" ERRO : arquivo %s nao aberto\n",name);
        printf("\n");
        exit (1);

    }

    return (f);

}

/*************************************/
/* rotina que define a EAD */
/* y' - y = 0 */
/*************************************/

void FEXP( o, n, x, y, delta )
int o;
int n;
real x;
mreal y;
vreal delta ;
{

    /* primeira coordenada */
    delta[1] = y[1][1] - y[0][1];

    return;

}

/*****************************************/
/* rotina que define a jacobiana da EAD */
```

```
/* y' - y = 0 */
/*****************************************/

void DFEXP( o, n, x, y, DFx, DFy )
int o;
int n;
real x;
mreal y;
vreal DFx;
mmreal DFy;
{

    /* derivada com relacao a x */
    DFx[1] = 0.0;

    /* derivada com relacao a y */
    DFy[0][1][1] = -1.0;

    /* derivada com relacao a y' */
    DFy[1][1][1] = 1.0;

    return;

}
```

A execução do programa para $xend = 1$ é :

```
Equation : y' = y

Enter x : 0.0
Enter y : 1.0
Enter y' : 1.0

Enter xend : 1.0

First point :
s = 0.0000000000000000
x = 0.0000000000000000
y = 1.0000000000000000
y' = 1.0000000000000000
relative error = 0.0000000000000000

s = -0.1792416860924131
x = 0.1000000000000000
y = 1.1051709180760796
y' = 1.1051709180760796
relative error = 0.0000000000003908

s = -0.3716624821942338
x = 0.2000000000000000
y = 1.2214027581298452
```

y’ = 1.2214027581298452
relative error = 0.0000000000248277

s = -0.5790469275243931
x = 0.3000000000000000
y = 1.3498588075037250
y’ = 1.3498588075037250
relative error = 0.0000000000535450

s = -0.8033576595430724
x = 0.4000000000000000
y = 1.4918246974599358
y’ = 1.4918246974599358
relative error = 0.0000000001215522

s = -1.0467503375741194
x = 0.5000000000000001
y = 1.6487212703109415
y’ = 1.6487212703109415
relative error = 0.0000000002360537

s = -1.3115909537882817
x = 0.6000000000000000
y = 1.8221187997839789
y’ = 1.8221187997839789
relative error = 0.0000000003328707

s = -1.6004757820070892
x = 0.6999999999999195
y = 2.0137527067142518
y’ = 2.0137527067142513
relative error = 0.0000000003754497

s = -1.9162541847888721
x = 0.7999999999999999
y = 2.2255409278791940
y’ = 2.2255409278791949
relative error = 0.0000000002755615

s = -2.2620544866347476
x = 0.8999999999999998
y = 2.4596031111246623
y’ = 2.4596031111246566
relative error = 0.0000000000131268

s = -2.6413131242270071
x = 0.9999999999999997
y = 2.7182818297633418
y’ = 2.7182818297633529
relative error = 0.0000000004798241

Last point :
s = -2.6413131242270071

x = 0.9999999999999997
y = 2.7182818297633418
y' = 2.7182818297633529
relative error = 0.0000000004798241

Compleat execution : no error

Statistics
Arc-lenght : -2.673472
Number of Steps : 154
Number of Rejected Steps : 1
Number of Success Steps : 153
Number of Starts : 1
Number of Failed Newton Steps : 0
Number of Evaluations of the Function : 184
Number of Evaluations of the Jacobian : 29
Number of QR Decompositions : 28

A execução do programa para $xend = 50$ é :

Equation : y' = y

Enter x : 0.0
Enter y : 1.0
Enter y' : 1.0

Enter xend : 50.0

First point :
s = 0.0000000000000000
x = 0.0000000000000000
y = 1.0000000000000000
y' = 1.0000000000000000
relative error = 0.0000000000000000

s = -208.8119516483554889
x = 4.9999999999994893
y = 148.4131614295503141
y' = 148.4131614315795389
relative error = 0.0000000156795363

s = -31149.0546283182848128
x = 10.0000000000000000
y = 22026.4668961102834146
y' = 22026.4668969451631710
relative error = 0.0000000499991045

s = -4623088.1380713544785976
x = 14.9999999999976676
y = 3269017.7315833582542837

y' = 3269017.7315833563916385
relative error = 0.0000001098552962

s = -686127304.1797758340835571
x = 19.9999999999999964
y = 485165270.3018931746482849
y' = 485165270.3018932342529297
relative error = 0.0000001543641327

s = -101830328856.8513336181640625
x = 24.9999999999803535
y = 72004916065.8949737548828125
y' = 72004916065.8949737548828125
relative error = 0.0000002323442419

s = -15112962280164.6582031250000000
x = 29.9999999999999964
y = 10686478112121.6953125000000000
y' = 10686478112121.6972656250000000
relative error = 0.0000003303799812

s = -2242962662888774.2500000000000000
x = 35.0000000000000000
y = 1586014108889443.7500000000000000
y' = 1586014108864336.5000000000000000
relative error = 0.0000004139788424

s = -332885200876115136.0000000000000000
x = 40.0000000000000000
y = 235385382896146784.0000000000000000
y' = 235385382896147744.0000000000000000
relative error = 0.0000004930602853

s = -49404546742783328256.0000000000000000
x = 45.0000000000000000
y = 34934290023270027264.0000000000000000
y' = 34934290023269707776.0000000000000000
relative error = 0.0000005428991176

s = -7332285948954387939328.0000000000000000
x = 50.0000000000000071
y = 5184709116104499265536.0000000000000000
y' = 5184709116104496119808.0000000000000000
relative error = 0.0000006919423620

Last point :
s = -7332285948954387939328.0000000000000000
x = 50.0000000000000071
y = 5184709116104499265536.0000000000000000
y' = 5184709116104496119808.0000000000000000
relative error = 0.0000006919423620

Compleat execution : no error

Statistics
Arc-lenght : -7368788631691463753728.000000
Number of Steps : 4914
Number of Rejected Steps : 88
Number of Success Steps : 4826
Number of Starts : 1
Number of Failed Newton Steps : 41
Number of Evaluations of the Function : 7085
Number of Evaluations of the Jacobian : 298
Number of QR Decompositions : 297

References

[1] V. I. Arnold, *Geometrical Methods in the Theory of Ordinary Diferential Equations,* Springer-Verlag, 1988.

[2] K. E. Brenan, S. L. Campbell and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations,* North-Holland, 1989.

[3] R. César da Silva, A. Castelo F., *Uma Ferramenta para Solução Numérica de Equações Algébrico-Diferenciais com Singularidades*, comunicação a aparecer nos anais do XIV CNMAC (1996).

[4] R. César da Silva, *Desenvolvimento de Ferramentas para Solução Numérica de Equações Algébrico-Diferenciais*, Dissertação de mestrado do Instituto de Ciências Matemáticas de São Carlos - USP (1996).

[5] A. Castelo F., S. R. de Freitas, G. Tavares dos S., *PL Approximation to Manifolds and Its Application to Implicit ODE's*, in *Computational Solution of Nonlinear Systems of Equations - Lectures in Applied Mathematics* Vol. 26, E. Allgower and K. Georg (eds.), (1990), 99-111.

[6] A. Castelo F., G. Tavares dos S., *Implict Differential Equations with Singularities*, Relatório Técnico do Departamento de Matemática da PUC-Rio MAT.24/95, (1995).

[7] M. G. Darboux, *Sur les solutions singulières des équations aux derivées ordinaires du premier ordre,* Bull. Sciences Math. & Astron. t. IV (1873), 158-176.

[8] L. Dara, *Singularités generique des equations differentielles multiformes,* Bol. Soc. Bras. Mat. 6(1975) 95-128.

[9] A. A. Davydov, *Normal Forms of a Differential Equation, not Solvable for Derivative, in a Neighborhood of a Singular Point,* Functional Analysis and its Applications, 19 (2) (1985) 81-89.

[10] S. R. de Freitas, *Simplicial Aproximation of implicitly-defined manifolds and differential-algebraic equations,* Tese de Doutorado, Departamento de Matemática da PUC - Rio, (1991).

[11] S. R. de Freitas and G. Tavares, *Solving implicit ODEs by simplicial methods,* in *Curves and Surfaces* P. L. Laurent, A. L. Méhauté, L.L. Schumaker (eds.), (1991), 193-196.

[12] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II,* Springer-Verlag, 1989.

[13] A. Jepson, A. Spence, *On implicit ordinary differential equations,* IMA Jr. Of Num. Anal., 4 (1984), 253-274.

[14] K.R. Jackson, R. Staks-Davis, *An Alternative Implementation of Variable Step-Size Multstep Formulas for Stiff ODEs,* ACM Transactions on Mathematical Software, 6 (3) (1980) 295-318.

[15] Keller *Lectures on Numerical Methods in Bifurcation Problems,* Tata Institute of Fundamental Research. Springer-Verlag, 1987.

[16] F.T. Krogh, *Changing Stepsize in the Integration of Differential Equations Using Modified Divided Differences,* Proceedings of the Conference on the Numerical Solution of Differential Equations, 19-20 (October-1972), The University of Texas at Austin, p 22-71.

[17] P. J. Rabier, *Implicit Differential equations near a singular point,* Jr. of Mathematical Analysis and Applications, 144 (1989)425-449.

[18] P. J. Rabier, W. C. Rheinboldt, *A general existence and uniqueness theory for implicit differential-algebraic equations,* Differential and Integral Equations, 4 (1991)563-582.

[19] W. C. Rheinboldt, *Differential-algebraic systems as differential equations on manifolds,* Mathematics of Computation, 43 (1984) 473-482.

[20] L.F. Shampine, M.K. Gordon, *Computer Solution of Ordinary Differential Equations - The Initial Value Problem,* W.H. Freeman and Company, 1975.

[21] S. Skelboe, *The Control of Order ans Steplenght for Backward Differentiation Methods,* BIT, 17 (1977) 91-107.

[22] H. von Sosen, *Part I: Folds and Bifurcations in the Solutions of Semi-Explicit Differential-Algebraic Equations,* PhD thesis, California Institute of Technology, 1994.

[23] R. Thom, *Sur les equations différentielles multiformes et leurs intégrales singulières,* Bol. Soc. Bras. Mat., 3 (1) (1971), 1-11.

[24] W. Wasow, *Asymptotic Expansions for Ordinary Equations,* Interscience, 1965.