**SOBRE MÉTODOS DE RUNGE-KUTTA
PARALELOS .**

César Guilherme de Almeida

Orientadora : Prof. Dra. Célia Maria Finazzi de Andrade

**Tese apresentada ao Instituto de Ciências
Matemáticas de São Carlos , para obtenção
do Título de Mestre em Matemática
Computacional .**

**São Carlos
1993**

AGRADECIMENTOS

Agradeço à professora Célia Maria Finazzi de Andrade por ser minha orientadora , mesmo estando aposentada .

Agradeço às professoras Cacilda Maria Ribeiro e Neide M. Bertoldi Franco pela ajuda que me deram na implementação dos métodos apresentados neste trabalho .

Agradeço às secretárias da Pós-Graduação do ICMSC , que sempre foram muito atenciosas comigo . Também agradeço ao pessoal do Laboratório de Sistemas Digitais (LASD) por me ajudarem a imprimir este trabalho .

Agradeço a minha esposa Claudia e aos meus filhos Gabriel e Janaína pela grande energia positiva que me transmitem , dando-me forças para realizar todos os meus trabalhos .

Finalmente agradeço ao CNPq e à FAPESP pelas bolsas de estudo concedidas e ainda agradeço à FAPESP pelo equipamento computacional que eu utilizei por intermédio de seu auxílio .

CONTEÚDO

Assunto	página
SÍNTESE	0
INTRODUÇÃO	1
CAPÍTULO 0	3
PRÉ-REQUISITOS	
0.1 - Métodos de Runge-Kutta : Principais Conceitos	3
0.2 - Noções Básicas de Teoria dos Grafos e Diferencial Elementar	7
CAPÍTULO 1	14
Teoria dos Métodos de Runge-Kutta Paralelos	
1.1 - Introdução	14
1.2 - Grafos Dirigidos e Métodos de Runge-Kutta	14
1.3 - Partição de Nível	16
1.4 - Alguns Métodos Com Potencial Para Paralelismo	20
1.4.1 - Métodos do Tipo I	22
1.4.2 - Métodos do Tipo II	34
1.5 - Conclusão	42
CAPÍTULO 2	43
Métodos de Runge-Kutta Implícitos Iterados Diagonalmente por um Processo Paralelo	
2.0 - Introdução	43
2.1 - Iteração Diagonal	43
2.1.1 - Ordem de Precisão	45
2.2 - Métodos PDIRK (RK Paralelo , Iterado Diagonalmente)	46
2.2.1 - Ordem de Métodos PDIRK	47
2.2.2 - Métodos PDIRK "Stiffly" Precisos	50
2.3 - Métodos PDIRK de Classes I e II	52
2.3.1 - Estabilidade de Métodos das Classes I e II	52
2.3.2 - Métodos com Matriz Diagonal de Elementos Iguais	53
2.3.3 - Métodos com Elementos Diagonais Arbitrários	61

2.4 - Métodos de Runge-Kutta Iterados até Convergência	65
2.4.1 - Métodos de Runge-Kutta com Elevada Ordem de Estágio	65
2.4.2 - Iteração Diagonal Aplicada a Métodos RK com Elevada Ordem de Estágio	67
2.5 - Conclusão	75
APÊNDICE	76
REFERÊNCIAS	77

ABSTRACT

This MSc dissertation presents a study on Parallel Runge-Kutta methods . It is based on some papers about the subject and contains two techniques for construction of Parallel methods . The first one uses the Elementary Differential Theory , while the second appeals to a Diagonal Iterated method for solving the implicit equations of the RK method .

Numerical results corresponding to both techniques are included . The computer language used was OCCAM , which is adequated to concurrent programming .

SÍNTESE

Esta dissertação de Mestrado apresenta um estudo referente a métodos de Runge-Kutta Paralelos .

Baseado em alguns artigos sobre o assunto , este trabalho , contém duas técnicas de construção de métodos Paralelos . Uma técnica é utilizar a Teoria das Diferenciais Elementares , a outra é recorrer ao processo iterativo diagonal para resolvermos as equações implícitas do método de Runge-Kutta .

Também apresentamos resultados numéricos , ou seja , implementamos os métodos construídos a partir de cada uma das técnicas . A linguagem utilizada foi a OCCAM , adequada para programação concorrente . Uma referência é o livro [Poutain,May,87] .

INTRODUÇÃO

Este trabalho analisa métodos de Runge-Kutta com potencial para paralelismo , isto é , métodos de Runge-Kutta que podem ser implementados em computadores que executam programas em paralelo . É necessário o conhecimento de uma linguagem apropriada para a computação paralela ; esta linguagem pode ser a OCCAM . Apresentamos alguns experimentos numéricos utilizando esta linguagem .

O texto está distribuído em três capítulos .

O capítulo zero é o inicial e é neste capítulo que damos os pré-requisitos necessários para se entender os conceitos referentes a métodos de Runge-Kutta . Estes métodos são analisados segundo a teoria das Diferenciais Elementares , que se relaciona perfeitamente com a teoria dos Grafos . Ambas as teorias são introduzidas neste capítulo inicial .

O capítulo um apresenta um estudo sobre potencialidade para paralelismo de métodos de Runge-Kutta e chegamos à conclusão de que as matrizes associadas a estes métodos devem ser esparsas . Duas famílias de métodos foram estudadas e suas matrizes são das formas abaixo :

$$A_1 = \begin{bmatrix} x & x & 0 & 0 \\ x & x & 0 & 0 \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix} , \quad A_{II} = \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & x & 0 & 0 \\ x & x & x & 0 \\ x & x & 0 & x \end{bmatrix} .$$

Definindo os grafos dirigidos , associados às matrizes acima , como sendo $G = (V,R)$, com

$$V = \{1,2,3,4\} \quad \text{e} \quad R = \{ (i,j) : i,j \in V \text{ e } a_{ij} \neq 0 \} ,$$

e considerando os diagramas correspondentes , obtemos esquemas que nos permitem analisar o potencial para paralelismo de cada um dos métodos que possuem estes tipos de matrizes .

Os métodos associados à matriz A_1 foram chamados de métodos do Tipo I e os métodos associados à matriz A_{II} foram chamados de métodos do Tipo II . Foram estudadas propriedades de estabilidade para ambos os métodos .

No capítulo dois fazemos um estudo de um processo iterativo diagonal aplicado a métodos de Runge-Kutta completamente implícitos . Este processo iterativo diagonal facilita a implementação de métodos completamente implícitos , porque o método iterativo obtido deste processo iterativo diagonal tem a mesma complexidade de um método diagonalmente implícito . Para analisarmos o potencial para paralelismo desses métodos iterativos , vamos escrever as suas equações implícitas como :

$$Y^{(k)} = h D F(Y^{(k)}) + y_{n-1} + h (A - D) F(Y^{(k-1)}) , \\ k = 1, 2, \dots ,$$

onde D é uma matriz diagonal arbitrária . Observamos que se considerarmos uma aproximação inicial $Y^{(0)}$ e fixarmos o número de iterações , então as equações implícitas que fornecem os valores de $Y^{(k)}$ podem ser resolvidas independente e simultaneamente , ou seja , podem ser implementadas em paralelo .

A eficiência dos métodos iterativos diagonais depende da escolha da matriz diagonal D ; pesquisamos algumas maneiras de se escolher a matriz D . Inicialmente estudamos métodos iterativos com um número fixo de iterações , com matrizes das formas : 1) $D = d I$, onde d é uma constante e I é a matriz identidade ; 2) $D = \text{diag} (d_1, \dots, d_s)$, com elementos diagonais arbitrários . Em ambos os casos conseguimos obter os parâmetros para construir métodos com boas propriedades de estabilidade . Outro processo que utilizamos para escolher a matriz diagonal D , quando o número de iterações não é fixado , foi o seguinte : exigimos que o raio espectral da matriz $I - D^{-1}A$ fosse zero . Também conseguimos construir métodos iterativos diagonais com boas propriedades de estabilidade , utilizando este processo de escolha da matriz D .

CAPÍTULO 0

PRÉ - REQUISITOS

0.1 - Métodos de Runge-Kutta : Principais Conceitos .

Neste capítulo daremos as noções básicas a respeito dos métodos de Runge-Kutta , que serão utilizados para resolver equações diferenciais ordinárias . Apresentaremos métodos explícitos e implícitos ; definiremos a ordem desses métodos e também alguns conceitos de estabilidade . Finalizaremos o capítulo apresentando a relação entre a Teoria das Diferenciais Elementares e as condições de ordem de métodos de Runge-Kutta .

Todos os conceitos e considerações apresentados podem ser encontrados em [Butcher,87] .

Seja o problema de valor inicial (PVI)

$$\begin{aligned} y'(t) &= f(y(t)) \quad , \quad t \in [t_0, T] , \\ y(t_0) &= y_0 , \end{aligned} \tag{0.1}$$

onde $y \in C^1(\mathbb{R}, \mathbb{R}^N)$ e $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$ é uma função contínua que satisfaz às condições de existência , unicidade e dependência contínua da solução em relação às condições iniciais .

Seja uma discretização de $[t_0, T]$ dada por : $t_n = t_0 + nh$, $n = 0, 1, 2, \dots$, onde h é o tamanho do passo . O método de Runge-Kutta de s -estágios (s -RK) , que fornece uma solução numérica para o problema (0.1) , é dado por :

$$\begin{aligned} Y_i &= y_{n-1} + h \sum_{j=1}^s a_{ij} f(Y_j) , \quad \text{para } i = 1, \dots, s , \\ y_n &= y_{n-1} + h \sum_{j=1}^s b_j f(Y_j) . \end{aligned} \tag{0.2}$$

Cada expressão Y_i dá uma aproximação para a função y calculada no ponto $t = t_{n-1} + c_i h$, $c_i = \sum_{j=1}^s a_{ij}$, para $i = 1, \dots, s$. O cálculo de y_n dá uma aproximação para $y(t_n)$.

Tendo em mente o método acima, vamos chamar de A a matriz de ordem s que possui elementos a_{ij} ; b e c denotarão vetores s -dimensionais com componentes b_i e c_i , respectivamente.

Definição 0.1. Um método de RK é explícito se e somente se $a_{ij} = 0$ para $i \leq j$. Um método que não é explícito é dito implícito.

No caso explícito, os valores de Y_i podem ser calculados recursivamente, sem a necessidade de resolver equações implícitas.

Uma classe de métodos implícitos importante é a classe dos métodos diagonalmente implícitos, definidos abaixo.

Definição 0.2. Um método de RK é diagonalmente implícito se e somente se $a_{ij} = 0$ para $i < j$. Um método é dito estritamente diagonalmente implícito se e somente se $a_{ij} = 0$ para $i \neq j$.

Observações :

1) Os métodos de RK implícitos que não são diagonalmente implícitos são chamados de completamente implícitos.

2) A vantagem dos métodos diagonalmente implícitos está no fato de que as equações implícitas dadas pelas expressões Y_i são mais simples que as equações de um método completamente implícito. De fato,

$Y_1 = y_{n-1} + h a_{11} f(Y_1)$ é implícita somente na incógnita Y_1 e conseqüentemente pode ser resolvida para Y_1 , sem envolver as incógnitas Y_j , $j > 1$. Similarmente, dados Y_j com $j \leq i - 1$, temos

que $Y_i = y_{n-1} + h \sum_{j=1}^i a_{ij} f(Y_j)$ é implícita somente na incógnita Y_i e conseqüentemente pode ser resolvida para Y_i , sem precisar envolver as incógnitas Y_j , com $j > i$.

3) No capítulo 2 exibiremos uma técnica de iteração diagonal que será usada para aproximar a solução de um método de Runge-Kutta completamente implícito. Fixaremos o número de iterações e mostraremos que o método resultante pertencerá à classe dos métodos de RK diagonalmente implícitos, com a vantagem de que suas equações implícitas poderão ser implementadas paralelamente, ou seja, usando programação concorrente (por exemplo a linguagem OCCAM), as equações implícitas serão resolvidas simultaneamente.

Definição 0.3. Se a matriz A de um método de Runge-Kutta implícito possuir um único auto valor, então o método será chamado de individualmente implícito.

Definição 0.4. Um método de Runge-Kutta possui ordem p se e somente se p é o maior inteiro tal que, para toda função f suficientemente diferenciável, o erro local satisfaz: $y_n - y(t_n) = O(h^{p+1})$ com $h \rightarrow 0$, onde $y(t)$ é a solução do PVI local:

$$y'(t) = f(y(t)), \quad y(t_{n-1}) = y_{n-1} \quad (0.3)$$

Definição 0.5. A função de estabilidade $R(z)$ de um método de Runge-Kutta é a função racional que satisfaz $y_n = R(z)y_{n-1}$, onde y_n é a aproximação gerada quando aplicamos o método de RK ao problema teste escalar $y'(t) = \lambda y(t)$, $\lambda \in \mathbb{C}$, e $z = \lambda h$, sendo h o tamanho do passo dado no método de RK.

Observação: Considerando $Y = (Y_1, \dots, Y_s)^t$, $e = (1, \dots, 1)^t$ vetores de dimensão s , o método (0.2), quando aplicado ao problema teste escalar visto acima, pode ser reescrito como:

$$\begin{aligned} Y &= y_{n-1} e + zAY \\ y_n &= y_{n-1} + zb^t Y \end{aligned}$$

Podemos isolar o vetor Y , na primeira equação acima, e substituí-lo na segunda equação, obtendo dessa forma:

$$y_n = [1 + zb^t(I - zA)^{-1}e]y_{n-1} = R(z)y_{n-1}.$$

Usando a identidade

$$1 + x^t N^{-1}y = \det(N + yx^t) / \det(N), \quad (0.4.0)$$

que pode ser provada usando a Regra de Crammer, podemos escrever a função de estabilidade como

$$R(z) = \frac{\det[(I - zA) + zeb^t]}{\det(I - zA)} = \frac{T(z)}{\det(I - zA)} \quad (0.4.1)$$

Definição 0.6. Um método de RK é chamado absolutamente estável em um ponto $z \in \mathbb{C}$, se, para este ponto z , $|R(z)| \leq 1$.

Definição 0.7. A região de estabilidade de um método de RK é o conjunto de pontos $z \in \mathbb{C}$ para os quais o método é absolutamente estável.

Definição 0.8. Um método de RK é A-estável se sua região de estabilidade contém $\mathbb{C}^-$ (conjunto de números complexos com parte real não positiva).

Definição 0.9. Um método de RK é fortemente A-estável se para todo ponto $z \in \mathbb{C}^-$, $|R(z)| < 1$.

Definição 0.10. Um método de RK é L-estável se for A-estável e ainda ocorrer

$$\lim_{|z| \rightarrow \infty} |R(z)| = 0.$$

Definição 0.11. Um método s-RK é algebricamente estável se as matrizes $B = \text{diag}(b_1, \dots, b_s)$ e $M = BA + A^t B - bb^t$ forem semi-definidas positivas.

0.2 - Noções Básicas de Teoria dos Grafos e Diferencial Elementar .

A relação entre Teoria dos Grafos e métodos de Runge-Kutta foi estudada por Butcher ([Butcher,87]) , que elaborou a Teoria das Diferenciais Elementares . Através desta teoria é possível obter um sistema de equações correspondente às condições de ordem de um método de Runge-Kutta , ou seja , para que um método atinja ordem p (definição 0.4) , basta resolvermos o sistema de equações concernente a este método . Este sistema é facilmente obtido com o auxílio de tipos especiais de grafos , os quais são chamados de árvores , nome sugerido pelos diagramas obtidos desses grafos .

Nesta seção vamos dar os conceitos básicos da Teoria dos Grafos e da Teoria das Diferenciais Elementares , vamos apresentar também algumas árvores seguidas de suas respectivas equações de ordem .

Definição 0.12 . Grafo dirigido ou di-grafo é um par (V,R) , onde V é um conjunto limitado e R é um conjunto de pares ordenados de membros distintos de V . Os membros de V são chamados de vértices e os membros de R são chamados de arcos .

Observação : Se (x,y) for um arco de um grafo dirigido (V,R) , então y será chamado de sucessor de x , e x será chamado de predecessor de y . O vértice que não possuir sucessor será chamado de terminal .

Definição 0.13 . Um grafo dirigido é dito injetivo se cada vértice possuir no máximo um predecessor .

Exemplo 1) Seja $V = \{x,y,z\}$ e $R = \{(x,y),(z,y)\}$, o diagrama do grafo dirigido (V,R) é dado por

(V,R) não é injetivo porque o vértice y possui dois predecessores .

Definição 0.14 . Um caminho em um grafo dirigido (V,R) é uma sequência de arcos distintos $r_1, \dots, r_n$, tais que $r_i = (v_{i-1}, v_i) \in R$ para $i = 1, \dots, n$, e $v_0, \dots, v_n \in V$. Tal caminho é de v_0 a v_n . Um ciclo é um caminho de um vértice a ele mesmo .

Definição 0.15 . Dois vértices v_1, v_2 são conectados por baixo se existir um caminho de v_1 a v_2 , ou de v_2 a v_1 , ou ainda, se existir um caminho de v_0 a v_1 e um de v_0 a v_2 , onde v_0 é um vértice qualquer em V . Um grafo dirigido é conectado por baixo se quaisquer dois vértices são conectados por baixo .

Exemplo 2) Seja $V = \{x,y,z\}$ e $R = \{(x,y)\}$, então o grafo dirigido (V,R) não é conectado por baixo, pois o vértice z não está conectado a nenhum outro vértice.

Definição 0.16 . Um grafo dirigido (V,R) é ordenado se existir uma ordem total sobre V , tal que, se (v_1, v_2) é um arco, então $v_1 < v_2$.

Exemplo 3) Seja $V = \{x,y,z,w\}$ e $R = \{(x,y), (y,z), (z,w), (w,x)\}$. O diagrama do grafo dirigido (V,R) é dado por

Este grafo dirigido possui um ciclo e portanto não pode ser ordenado, porque isto ocasionaria $x < x$.

Definição 0.17 . Uma árvore é um grafo dirigido injetivo, ordenado e conectado por baixo. A raiz da árvore é dada pelo ínfimo do conjunto de todos os vértices.

Exemplo 4) Daremos abaixo os diagramas de árvores com até três vértices.

Exemplo 5) A árvore com um único vértice será denotada por τ . Quando desconectamos a raiz dos demais vértices de uma árvore obtemos sub-árvores, então podemos usar a seguinte notação para escrevermos uma árvore: $t = [t_1 t_2 \dots t_m]$, onde $t_1, \dots, t_m$ são sub-árvores da árvore t . Se considerarmos o exemplo 4), então temos as seguintes notações:

Árvores	Notações
τ
$[\tau]$
$[\tau \tau]$
$[[\tau]]$

Definiremos agora algumas funções concernentes às árvores e também definiremos formalmente Diferencial Elementar e Peso Elementar.

Definição 0.18. A ordem de uma árvore t é o número de vértices que esta árvore possui. A função recursiva $r(t)$, definida abaixo, dá a ordem de uma árvore genérica t :

$$r(\tau) = 1, \quad r([t_1 \dots t_m]) = 1 + r(t_1) + \dots + r(t_m).$$

Exemplo 6) Considerando as notações usadas no exemplo 5), vemos facilmente que a função r dá as ordens das árvores daquele exemplo:

$$r(\tau) = 1,$$

$$r([\tau]) = 1 + r(\tau) = 2,$$

$$r([\tau \tau]) = 1 + r(\tau) + r(\tau) = 3,$$

$$r[[\tau]] = 1 + r([\tau]) = 1 + 2 = 3.$$

Definição 0.19 . Dada uma árvore t , definimos a função densidade $\gamma(t)$, recursivamente , como segue abaixo

$$\gamma(r) = 1 \quad , \quad \gamma([t_1 \dots t_m]) = \gamma([t_1 \dots t_m])\gamma(t_1)\dots\gamma(t_m)$$

Observação : existe uma maneira prática de se obter o valor da função densidade : cada vértice v da árvore t será rotulado com o inteiro i , onde i é a ordem da sub-árvore que possui v como raiz . Assim , é claro que os vértices terminais serão rotulados com o inteiro 1 . O valor da densidade de t será o produto dos números que rotulam t .

Exemplo 7) Consideremos a árvore $t = [r [1]]$ que possui o seguinte diagrama :

A função densidade da árvore t assume o valor 8 , facilmente calculado :

$$\gamma(t) = \gamma([r [1]]) = 4 \times 2 \times 1 \times 1 = 8 .$$

Na definição a seguir usamos o conhecido funcional linear projeção $\Pi_j : \mathbb{R}^N \rightarrow \mathbb{R}^N$ dado por :

$$\Pi_j(x) = x_j ,$$

onde $x = (x_1, \dots, x_N) \in \mathbb{R}^N$.

Definição 0.20 . Dadas uma árvore t e uma função $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$, suficientemente diferenciável , a Diferencial Elementar associada à árvore t é definida recursivamente como a função $F(t) : \mathbb{R}^N \rightarrow \mathbb{R}^N$, tal que , se $x \in \mathbb{R}^N$, então :

$$\text{se } t = r \quad , \quad F(t)(x) = f(x) ;$$

$$\text{se } t = [t_1 \dots t_m] \quad , \quad F(t)(x) = f^{(m)}(x)(F(t_1)(x), \dots, F(t_m)(x)) ,$$

onde $f^{(m)}(x)$ é um operador m -linear .

Para $i = 1, \dots, N$, a i -ésima componente do operador $f^{(m)}(x)$, aplicada a um ponto $(y_1, \dots, y_m)$, pertencente a $\mathbb{R}^N \times \dots \times \mathbb{R}^N$, é dada por :

$$\sum_{j_1=1}^N \dots \sum_{j_m=1}^N \partial_{j_1 \dots j_m} f_i(x) u_{j_1}(y_1) \dots u_{j_m}(y_m) ,$$

onde $f_i(x) = (\pi_i \circ f)(x)$.

Exemplo 8) Vamos considerar uma função diferenciável $f : \mathbb{R} \Rightarrow \mathbb{R}$. Então, para $t = [\tau]$ a Diferencial Elementar associada a esta árvore é dada por $F(t) : \mathbb{R} \Rightarrow \mathbb{R}$, tal que, se $x \in \mathbb{R}$, então

$$F(t)(x) = f^{(2)}(x)(F(\tau)(x), F(\tau)(x)) = f^{(2)}(x)(f(x), f(x)) .$$

Neste exemplo temos o seguinte operador bi-linear :

dados $y_1, y_2 \in \mathbb{R}$,

$$f^{(2)}(x)(y_1, y_2) = \partial_{11} f(x) y_1 y_2 = f''(x) y_1 y_2$$

onde $f''(x)$ é a derivada segunda da função f , calculada no ponto x .

Definição 0.21 . Consideremos o método s-RK (0.2) . Dada uma árvore t , o Peso Elementar $Q_t(t)$, correspondente ao estágio i , $i = 1, \dots, s$, é definido de maneira recursiva como segue :

$$\text{se } t = \tau, Q_t(t) = c_i ;$$

$$\text{se } t = [t_1 \dots t_m], Q_t([t_1, \dots, t_m]) = \sum_{j=1}^s a_{ij} Q_{j_1}(t_1) \dots Q_{j_m}(t_m) .$$

O Peso Elementar da árvore t é dado por $Q(t) = Q_{s+1}(t)$, se considerarmos a seguinte notação

$$b_j = a_{s+1,j}, j = 1, \dots, s \text{ e } c_{s+1} = \sum_{j=1}^s b_j .$$

Observação : existe uma maneira de se rotular uma árvore de modo que o Peso Elementar associado a esta árvore seja facilmente obtido . Dada uma árvore t , nós rotulamos a sua raiz com o índice i , e os demais vértices são rotulados com outros índices , por exemplo j , k , l , etc . Então o Peso Elementar é obtido fazendo-se o somatório :

$\sum b_{i,j} a_{i,j} \dots$, sobre todos os índices i , j , k , l , etc , que variam de 1 a s se o método de Runge-Kutta tiver s estágios . Se dois índices i_1 e i_2 , que fazem parte da rotulação , formam um arco da árvore t , então o coeficiente $a_{i_1 i_2}$ faz parte do somatório acima .

Exemplo 9) Dada a árvore $t = [1]$, vamos obter o Peso Elementar , associado a ela , usando a rotulação explicada acima .

Árvore

|

Rotulação

|^j_i

Peso Elementar

$$\sum_{i,j=1}^s b_{i,j} a_{i,j} = \sum_{i=1}^s b_{i,i} c_i$$

Daremos agora o teorema que relaciona a teoria das Diferenciais Elementares com a ordem de um método de Runge-Kutta de s estágios .

Teorema 0.1 . Um método s - RK possui ordem p se e somente se $Q(t) = \frac{1}{\gamma(t)}$, para toda árvore t , tal que $r(t) \leq p$.

A prova deste teorema encontra-se em [Butcher,87] , p. 172 .

■

Exemplo 10) Vamos exibir uma tabela com as árvores de até quatro vértices com os seus respectivos valores de $Q(t)$ e $\gamma(t)$. (Os somatórios que aparecem aqui têm todos os seus índices variando de 1 a s , se o método de RK tiver s estágios .)

árvore	Peso Elementar(Q(t))	$\gamma(t)$
$\sum b_i$	1
$\sum b_i c_i$	2
$\sum b_i c_i^2$	3
$\sum b_i a_{ij} c_j$	6
$\sum b_i c_i^3$	4
$\sum b_i c_i a_{ij} c_j$	8
$\sum b_i a_{ij} c_j^2$	12
$\sum b_i a_{ij} a_{jk} c_k$	24

Observação : No capítulo 2 nós introduziremos o conceito de ordem de estágio de um método de Runge-Kutta implícito . Usando teoria das Diferenciais Elementares , Butcher também conseguiu exibir um sistema de equações que fornece exatamente as condições de ordem de estágio , ou seja , se resolvermos o sistema de equações , então obtemos a ordem de estágio desejada . Butcher chamou este sistema de equações de Hipóteses Simplificadoras de Linha .

CAPÍTULO 1

Teoria dos Métodos de Runge-Kutta Paralelos

1.1 - Introdução .

Neste capítulo pretendemos estudar a potencialidade para paralelismo existente nos métodos de Runge-Kutta (RK) , ou seja , queremos estudar as possibilidades de implementar tais métodos usando um computador que executa programas em paralelo . Lembramos que , para este objetivo ser atingido , é necessário usarmos uma linguagem apropriada , como por exemplo , a OCCAM .

A potencialidade dos métodos RK será explorada através da análise da estrutura das matrizes correspondentes a esses métodos . Com o auxílio da Teoria dos Grafos poderemos determinar quais as matrizes adequadas para o nosso estudo .

Vamos nos concentrar na análise de duas famílias de métodos semi-implícitos de quatro estágios , ordem quatro e que utilizem dois processadores em suas implementações . A base dessa análise será a Teoria das Árvores .

Os estudos aqui expostos foram desenvolvidos por A. Iserles e S.P. Norsett e encontram-se em [Iserles,Norsett,90] .

1.2 - Grafos Dirigidos e Métodos de Runge-Kutta .

Consideremos novamente o PVI autônomo (0.1) e o método de RK (0.2) .

Os elementos a_{ij} e b_j são parâmetros que definem o método e é usual a representação matricial :

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \hline c_s & a_{s1} & \dots & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array} \quad , \quad \text{ou} \quad \begin{array}{c|c} c & A \\ \hline & b^t \end{array} \quad (1.1)$$

, onde $c_i = \sum_{j=1}^s a_{ij}$.

Para investigarmos a potencialidade para paralelismo neste método , nós introduzimos o di-grafo (grafo dirigido) da matriz A , dado por : $G = (V,R)$, onde

$V = \{1, \dots, s\}$ é o conjunto de vértices e

$R = \{(i,j) : i,j \in V \text{ e } a_{ij} \neq 0\}$ é o conjunto de arcos .

Notemos que nesta definição de di-grafo nós permitimos que arcos da forma (i,i) pertençam a R , contudo sempre ficará claro no contexto quando isto ocorrer .

Usaremos a seguinte convenção : se (i,j) é um arco , então a flecha que une estes dois vértices aponta para i . Isto é feito para explicitar as relações entre grafo dirigido e implementação paralela .

Definição 1.1 : Dada a matriz A e seu di-grafo definimos :

- o vértice v constitui um 1-loop se $a_{vv} \neq 0$ e ele será um 0-loop se $a_{vv} = 0$;
- para $m > 1$, um m-loop é um ciclo abrangendo m vértices distintos .

Exemplo 1 : Suponhamos que $V = \{1,2\}$ e que a matriz A seja da forma :

$$A = \begin{bmatrix} x & x \\ x & 0 \end{bmatrix} ,$$

onde x indica um elemento distinto de zero . Se considerarmos dois vértices $v_1 = 1$ e $v_2 = 2$, obtemos um 2-loop , com o ciclo que relaciona os elementos a_{12} e a_{21} da matriz A . Se considerarmos um único vértice , obtemos um 1-loop com o vértice $v = 1$ e um 0-loop com o vértice $v = 2$. O di-grafo de A , neste exemplo , é representado pelo diagrama abaixo .

Definição 1.2 : Um método de Runge-Kutta é chamado σ -implícito quando σ for o comprimento do maior loop encontrado no di-grafo da matriz correspondente a este método .

Observação : Usando a terminologia acima , podemos dizer que um método explícito corresponde a um método de Runge-Kutta 0-implícito , e que um diagonalmente implícito corresponde a um método 1-implícito .

Exemplo 2 : Dada uma matriz

$$A = \begin{bmatrix} 0 & x & x \\ x & 0 & x \\ x & x & 0 \end{bmatrix} ,$$

o diagrama correspondente ao seu di-grafo é dado por

Considerando-se três vértices podemos obter um 3-loop com o ciclo que relaciona os elementos a_{12} , a_{23} e a_{31} da matriz A . Dessa forma , vemos que neste exemplo $\sigma = 3$.

1.3 - Partição de Nível .

Vamos assumir que o conjunto de vértices $\{1, \dots, s\}$ é particionado em v subconjuntos disjuntos e não vazios , ou seja ,

$$\begin{aligned} \{1, \dots, s\} &= \bigcup_{i=1}^v P_i ; \quad P_1, \dots, P_v \neq \emptyset \\ &\text{e } P_i \cap P_j = \emptyset , \quad \forall i \neq j . \end{aligned} \tag{1.2}$$

Nós chamamos (1.2) uma partição em processadores . Agora , fazemos a partição de cada P_l :

$$P_l = \bigcup_{i=1}^u L_{l,i} , \quad l = 1, \dots, v , \tag{1.3}$$

onde $L_{l,j} \cap L_{l,i} = \emptyset , \quad \forall i \neq j$.

Notemos que os $L_{l,i}$ podem ser conjuntos vazios . Contudo , assumimos que , dado $r \in \{1, \dots, v\}$, é verdade que $L_{r,j} \neq \emptyset$, para algum $j = 1, \dots, u$; $u \geq 1$.

Nós dizemos que (1.3) é uma partição de nível , associada a um método , se valerem as seguintes condições :

a) para cada $l = 1, \dots, v$ e $j = 1, \dots, u$, todos os vértices em $L_{l,j}$ pertencem a um loop ;

b) se $a_{ij} \neq 0$, então :

ou existem $p \in \{1, \dots, v\}$ e $r, w \in \{1, \dots, u\}$ tais que $r \leq w$, $i \in L_{p,w}$ e $j \in L_{p,r}$;

ou existem $q, t \in \{1, \dots, v\}$ e $r, w \in \{1, \dots, u\}$ tais que $q \neq t$, $r < w$, $i \in L_{t,w}$ e $j \in L_{q,r}$.

Uma combinação de uma partição de nível e uma partição em processadores , de um método σ -implícito , será chamada uma implementação $\{v, u, \sigma\}$.

O significado intuitivo da partição de nível diz respeito à comunicação entre os níveis de P_l ($l = 1, \dots, v$) . Em outras palavras , os vértices em cada nível $L_{l,i}$ devem se "comunicar" entre si neste conjunto , ou seja , estão num loop ; porém não se "comunicam" com os vértices nos demais $L_{k,i}$ ($k \neq l$) , isto é , não estão conectados com os vértices em $L_{k,i}$. Entretanto , podem se comunicar com os vértices em $L_{k,j}$, com $j > i$. Sendo esta comunicação só no sentido de "enviar informações" (as flechas que aparecem no diagrama do di-grafo apontam para um único sentido , neste caso) .

Exemplo 3 : Dada a matriz abaixo

$$A = \begin{bmatrix} x & x & 0 & 0 \\ x & x & 0 & 0 \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix} ,$$

o diagrama de seu di-grafo é dado por :

É fácil ver que neste caso $\sigma = 2$, e o método de Runge-Kutta , correspondente a esta matriz , é 2-implícito . Também é fácil notar que este método possui implementação $\langle 2,1,2 \rangle$, isto é , são usados dois processadores , cada um com um único nível , e cada um desses níveis está encarregado de calcular as duas relações implícitas de cada bloco diagonal 2×2 . Esta observação é muito importante , porque ela nos mostra que um método de 4 estágios pode ser tratado como dois métodos de dois estágios cada um .

Teorema 1.1 : A ordem p de um método de Runge-Kutta explícito com implementação $\langle v,u,0 \rangle$ não pode exceder u , independente do número de processadores v .

Prova : Suponhamos por absurdo que a ordem exceda u . Então , pelo Teorema (0.1) , a equação de ordem correspondente a árvore linear $t = [u]_u$, que possui o diagrama abaixo :

é dada por $Q(t) = 1/\gamma(t)$. Usando as definições de Peso Elementar e da função γ , a igualdade acima é equivalente a :

$$b^t A^{u-1} c = 1/(u+1)! \quad , \quad (1.4)$$

onde A , b , c correspondem à notação em (1.1) .

Provaremos que o lado esquerdo de (1.4) é nulo , obtendo uma contradição .

Seja $g_l := A^l c$ ($l = 0, 1, \dots$). Para todo $q \in V$ tal que $i \in L_{q,1}$, é verdade que $(g_0)_i = c_i = 0$. De fato, como $c_i = \sum_{j=1}^{i-1} a_{ij}$ (método explícito), então se $a_{ij} \neq 0$ para algum j , pela definição de partição de nível, teríamos que $j \in L_{q,1}$, ou $j \in L_{t,r}$, com $t \neq q$ e $r < 1$. Como esta última possibilidade não ocorre, então $j \in L_{q,1}$, e dessa forma $i, j \in L_{q,1}$, o que é uma contradição, pois pela definição de partição de nível - item a) - juntamente com o fato do método ser explícito ($\sigma = 0$), temos que cada nível $L_{t,i}$ tem no máximo um elemento, portanto $a_{ij} = 0$ e conseqüentemente $c_i = 0$.

Agora, provaremos por indução sobre k que, se existirem $q \in V$ e $r \in \{1, \dots, k+1\}$ tais que $i \in L_{q,r}$, então necessariamente $(g_k)_i = 0$.

A afirmação é válida para $k = 0$. Vamos assumi-la verdadeira para $k = m-1$ e prová-la para $k = m$. É fácil ver que $g_m = A g_{m-1}$, logo segue que

$$(g_m)_i = \sum_{j=1}^s a_{ij} (g_{m-1})_j \quad (1.5)$$

Suponhamos que $i \in L_{t,l}$ para algum $l \in \{1, \dots, m+1\}$, e que $a_{ij} \neq 0$ para algum $j \in \{1, \dots, s\}$; então existe $q \in V$, tal que, $j \in L_{q,r}$ com $r \in \{1, \dots, l-1\}$, e pela hipótese de indução $(g_{m-1})_j = 0$. Se $j \in L_{t,r}^-$ com $l \leq r \leq m+1$, então para $i \in L_{t,l}$, temos que $a_{ij} = 0$. Logo, no somatório (1.5), ora a_{ij} é zero, ora $(g_{m-1})_j$ é zero, para todo $j \in \{1, \dots, s\}$, e assim nossa afirmação é verdadeira para $k = m$ e portanto para todo k natural.

Vamos assumir $k = u-1$, logo, para todo $i \in \{1, \dots, s\}$, existem $q \in V$ e $r \in \{1, \dots, u\}$ tais que $i \in L_{q,r}$. Portanto g_{u-1} é o vetor nulo e assim

$$b^t A^{u-1} c = b^t g_{u-1} = 0,$$

o que contradiz (1.4).

Sabemos que , para $1 \leq s \leq 4$, existem métodos s-RK explícitos que possuem ordem s , então implementação paralela não pode aumentar a razão ordem/estágio , para tais métodos . No caso de métodos com $s \geq 5$, a ordem é sempre menor que o número de estágios e é possível obtermos algum melhoramento via implementação paralela . Entretanto , este melhoramento não é substancial , sendo mais compensador utilizar métodos implícitos .

Exemplo 4 : Seja o método 6-RK com ordem 5 ; [Butcher,87] , p. 199 .

0	0					
1/8	1/8	0				
1/4	0	1/4	0			
1/2	-1/2	1	0	0		
3/4	21/16	-15/8	3/4	9/16	0	
1	-17/7	4	0	-12/7	8/7	0
	7/90	0	32/20	12/90	32/90	7/90

É fácil observar que este método admite partição de nível $\{2,5,0\}$:

$$L_{11} = \{1\} , L_{21} = \emptyset ,$$

$$L_{12} = \{2\} , L_{22} = \emptyset ,$$

$$L_{13} = \{3\} , L_{23} = \{4\} ,$$

$$L_{14} = \{5\} , L_{24} = \emptyset ,$$

$$L_{15} = \{6\} , L_{25} = \emptyset .$$

Portanto , usar dois processadores é mais eficiente que usar um , porque podemos usar $u = 5$ estágios sequenciais ao invés dos 6 estágios usados na programação sequencial .

1.4 - Alguns Métodos Com Potencial Para Paralelismo .

Vamos passar a estudar duas famílias de métodos de Runge-Kutta , cada uma delas composta por métodos que chamaremos do Tipo I e do Tipo II , respectivamente .

Dada uma matriz A associada a um método s-RK , diremos que este método é do Tipo I se :

$$a1) s = \hat{u}v ;$$

$$b1) A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_v \end{bmatrix}$$

onde cada uma das matrizes $A_1, \dots, A_v$ é $\hat{u} \times \hat{u}$;

e ele é do Tipo II se :

$$a2) s = \hat{u}v ;$$

b2) a matriz A possui a forma bloco-triangular-inferior :

$$A = \begin{bmatrix} \tilde{A} & 0 & \dots & 0 \\ D_{21} & \tilde{A} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ D_{\hat{u}1} & \dots & D_{\hat{u},\hat{u}-1} & \tilde{A} \end{bmatrix} ,$$

onde $\tilde{A}$, D_{ij} ($i = 2, \dots, \hat{u}$; $j = 1, \dots, i-1$) e 0 são matrizes $v \times v$, $\tilde{A}$ é diagonal e 0 é a matriz nula .

Observação : A matriz do exemplo 3 , subseção 1.3 , é de um método do Tipo I , com $\hat{u} = 2$ e $v = 2$; já um método do Tipo II , com estes mesmos valores de $\hat{u}$ e v , possui matriz dada por :

$$A = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ d_1 & c_2 - d_1 - c_1 & c_1 & 0 \\ c_4 - c_2 - d_2 & d_2 & 0 & c_2 \end{bmatrix} \quad (1.6)$$

É fácil ver que o diagrama do grafo dirigido associado à matriz acima é dado por :

Este esquema sugere que podemos implementar o método do Tipo II usando dois processadores , só que estes precisam se comunicar entre si . Para explicar melhor como funciona esta comunicação , precisamos lembrar que um método 4-RK implícito possui quatro equações implícitas , que fornecem Y_1 , Y_2 , Y_3 e Y_4 . Estas equações são resolvidas da seguinte maneira : as duas primeiras equações , que fornecem Y_1 e Y_2 , são resolvidas independente e simultaneamente em cada um dos processadores . As equações para Y_3 e Y_4 dependem dos valores de Y_2 e Y_1 respectivamente , então o canal de comunicação entre os dois processadores funciona da seguinte forma : o processador que fornece o valor de Y_1 recebe o valor de Y_2 e o outro processador recebe o valor de Y_1 . Feita esta comunicação , a terceira e a quarta equações , que fornecem Y_3 e Y_4 , são resolvidas independente e simultaneamente em cada um dos seus processadores correspondentes .

1.4.1 - Métodos do Tipo I .

Vamos estudar métodos do Tipo I que tenham implementação $\{v,1,2\}$. Provaremos que a ordem máxima de tais métodos é quatro , independente do número v de processadores , dessa forma , nos concentraremos no estudo de métodos que possuem implementação $\{2,1,2\}$. Estes métodos podem ser escritos como soma de dois métodos 2-RK , ditos internos , e usaremos a seguinte representação :

$$\begin{array}{c|c} c_1 & A_1 \\ \hline \rho_1 & (b^{(1)})^t \end{array} + \begin{array}{c|c} c_2 & A_2 \\ \hline \rho_2 & (b^{(2)})^t \end{array} , \quad (1.7a)$$

onde c_k , $b^{(k)}$, $k = 1,2$, são vetores bi-dimensionais e A_k é uma matriz 2×2 . A aproximação y_n no ponto t_n , no final de cada passo , é obtida efetuando-se :

$$y_n = \sum_{k=1}^2 \rho_k y_n^{(k)} ,$$

onde $y_n^{(k)}$ é a aproximação interna dada por

$$y_n^{(k)} = y_{n-1} + h \sum_{i=2k-1}^{2k} b_i^{(k)} f(Y_i) , \quad k = 1, 2 .$$

Observações :

- 1) o vetor $b^{(k)}$ está associado a um vetor b através da expressão $\rho_k b_i^{(k)} = b_{2(k-1)+i}$, $i = 1, 2$; $k = 1, 2$, onde b é o vetor correspondente ao método de Runge-Kutta que possui matriz :
- $A = \text{diag}(A_1, A_2)$.

(O índice usado abaixo de um vetor representa a sua componente)

- 2) Se os métodos internos e o original forem consistentes , ou seja ,

$$\sum_{i=1}^2 b_i^{(k)} = 1 \quad \text{e} \quad \sum_{j=1}^4 b_j = 1 , \quad \text{então obtemos que}$$

$$\sum_{k=1}^2 \rho_k = 1 . \quad (1.7b)$$

- Análise de Ordem .

Nesta subseção vamos analisar condições de ordem para um método do Tipo I com implementação $\{2,1,2\}$, as quais são extraídas da Teoria das Diferenciais Elementares , que introduzimos no capítulo 0 .

Consideremos que os métodos internos , vistos acima , possuam ordem três . Vamos escrever suas matrizes como :

$$A_k = \begin{bmatrix} a_1^{(k)} & c_1^{(k)} - a_1^{(k)} \\ c_2^{(k)} - a_2^{(k)} & a_2^{(k)} \end{bmatrix} , \quad k = 1, 2 .$$

(Com o objetivo de não sobrecarregar a notação usaremos os elementos do k-ésimo método interno sem os índices k , e quando for conveniente voltaremos a usá-los) .

Segundo o Teorema 0.1 , ordem três requer a solução das equações abaixo .

$$\sum_{i=1}^2 b_i = 1 \quad , \quad \sum_{i=1}^j b_i c_i = 1/2 \quad , \quad \sum_{i,j=1}^2 b_i a_{ij} c_j = 1/6 \quad ,$$

e

$$\sum_{i=1}^2 b_i c_i^2 = 1/3 \quad ,$$

ou equivalentemente a solução das equações :

$$b_1 = (c_2 - \frac{1}{2}) / (c_2 - c_1) \quad ;$$

$$b_2 = - (c_1 - \frac{1}{2}) / (c_2 - c_1)$$

e

$$c_1 c_2 - \frac{1}{2}(c_1 + c_2) + \frac{1}{3} = 0 \quad ;$$

$$c_1 c_2 - a_1 (c_2 - \frac{1}{2}) - a_2 (c_1 - \frac{1}{2}) = \frac{1}{6} \quad .$$

Definindo : $d = c_1 + c_2 - 1$ e $q = a_1 + a_2$, as duas últimas relações ficam :

$$c_1 c_2 = \frac{1}{2} d + \frac{1}{6} \quad , \quad (1.8a)$$

$$a_1 c_2 + a_2 c_1 = \frac{1}{2} (d + q) \quad . \quad (1.8b)$$

As condições de ordem quatro para o método combinado (1.7a) , segundo o Teorema 0.1 , são equivalentes às equações :

$$\sum_{k=1}^2 \rho_k Q^k(t) = \frac{1}{\gamma(t)} \quad ,$$

com t representando , em cada uma das equações , uma árvore de até quatro vértices , e $Q^k(t)$ representa o Peso Elementar , do k -ésimo método interno , correspondente a essa árvore t . Lembremos que no capítulo 0 nós exibimos , na tabela do exemplo 10 , as condições de ordem quatro utilizando as árvores de até quatro vértices . Agora , vamos exibir uma tabela com quatro árvores , cada uma delas contendo quatro vértices . Ao lado de cada uma dessas árvores estão as suas respectivas equações de ordem , já alteradas devido aos resultados obtidos antes de (1.8) .

Tabela 1.1 : árvores de 4 vértices e suas equações de ordem .

Árvore	Equação de Ordem
a)	$\frac{1}{2} c_1 c_2 + (c_1 - \frac{1}{2}) a_1 c_2 + (c_2 - \frac{1}{2}) a_2 c_1 -$ $-(c_2 - \frac{1}{2}) a_1^2 - (c_1 + c_2 - 1) a_1 a_2 - (c_1 - \frac{1}{2}) a_2^2 = \frac{1}{24}$
b)	$-(c_2 - \frac{1}{2})(c_1 + c_2) a_1 - (c_1 - \frac{1}{2})(c_1 + c_2) a_2 +$ $-(\frac{1}{2}) c_1 c_2 + c_1 c_2 (c_1 + c_2) = \frac{1}{12}$
c)	$\frac{1}{2} c_1 c_2 - c_1 (c_2 - \frac{1}{2}) a_1 - (c_1 - \frac{1}{2}) c_2 a_2 = \frac{1}{8}$
d)	$- c_1 c_2 (c_1 + c_2) + \frac{1}{2} (c_1^2 + c_1 c_2 + c_2^2) = \frac{1}{4}$

Vemos que as equações de ordem referentes aos itens a) e c) são ambas equivalentes a :

$$\sum_{k=1}^2 \rho_k q_k = \frac{1}{2} , \quad (1.9)$$

onde $q_k = a_1^{(k)} + a_2^{(k)}$; já as equações referentes aos itens b) e d) são ambas equivalentes a :

$$\sum_{k=1}^2 \rho_k d_k = 0 \quad , \quad (1.10)$$

onde $d_k = c_1^{(k)} + c_2^{(k)} - 1$.

Teorema 1.2 : Métodos do Tipo I , com implementação $\{v,1,2\}$, não atingem ordem cinco independente da escolha de v .

Prova : A prova que daremos a seguir considerará a variação do índice k de 1 a 2 , sem perda de generalidade , pois a alteração desse índice não modifica a estrutura da demonstração . Será necessário o conhecimento de duas equações de ordem referentes a duas árvores de cinco vértices , as quais estão dadas na tabela abaixo .

Tabela 1.2 : Duas árvores de cinco vértices e suas equações de ordem

Árvore	Equações de Ordem
e)	$b_1 c_1^2 [c_1 c_2 - (c_2 - c_1) a_1] +$ $+ b_2 c_2^2 [c_1 c_2 + (c_2 - c_1) a_2] = \frac{1}{10}$
f)	$b_1 [c_1^3 c_2 + c_1 c_2 (c_2 - c_1) a_1 + c_1 (c_2^2 - c_1^2) a_2 +$ $- (c_2^2 - c_1^2) a_1^2 - (c_2^2 - c_1^2) a_1 a_2] +$ $+ b_2 [c_1 c_2^3 - c_2 (c_2^2 - c_1^2) a_1 - c_1 c_2 (c_2 - c_1) a_2 +$ $+ (c_2^2 - c_1^2) a_2^2 + (c_2^2 - c_1^2) a_1 a_2] = \frac{1}{60} \quad .$

Resolvendo o sistema linear

$$\begin{aligned} a_1 + a_2 &= q \\ a_1 c_2 + a_2 c_1 &= \frac{1}{2} (d + q) \quad , \end{aligned}$$

obtemos

$$a_1 = \left(\frac{1}{2} (d + q) - c_1 q \right) / (c_2 - c_1) , \quad (1.11)$$

$$a_2 = \left(\frac{1}{2} (d + q) - c_2 q \right) / (c_1 - c_2)$$

Substituindo os valores de a_1 e a_2 , b_1 e b_2 , juntamente com a fórmula (1.8a), na equação do item e) da tabela 1.2, obtemos:

$$Q_e^{(k)} = \frac{1}{12} (d_k + 1) q_k + \frac{1}{18} , \quad (1.12)$$

onde $Q_e^{(k)}$ representa o Peso Elementar do k -ésimo método interno referente à árvore do item e) da tabela 1.2. Sabendo que vale (1.7b) e que a condição de ordem para a árvore do item e) é dada por:

$$\sum_{k=1}^2 \rho_k Q_e^{(k)} = \frac{1}{10} ,$$

usando a fórmula (1.9) obtemos:

$$\sum_{k=1}^2 \rho_k d_k q_k = \frac{1}{30} . \quad (1.13)$$

O Peso Elementar associado à árvore do item f) da tabela 1.2 pode ser reescrito como

$$Q_f = c_1 c_2 (b_1 c_1^2 + b_2 c_2^2) + (c_2 - c_1) (b_1 c_1 [c_2 a_1 + (c_1 + c_2) a_2] + \\ - b_2 c_2 [(c_1 + c_2) a_1 + c_1 a_2]) + (c_2^2 - c_1^2) (b_2 a_2 - b_1 a_1) (a_1 + a_2)$$

Usando $\sum_{i=1}^2 b_i c_i^2 = \frac{1}{3}$ e $q = a_1 + a_2$ segue:

$$Q_f = \frac{1}{3} c_1 c_2 + (c_2 - c_1) [c_1 c_2 (b_1 - b_2) q + (b_1 c_1^2 a_2 - b_2 c_2^2 a_1)] + \\ + q (c_2^2 - c_1^2) (b_2 a_2 - b_1 a_1) .$$

Usando (1.8a), (1.11) e sabendo que $d + 1 = c_1 + c_2$,

$$\sum_{i=1}^2 b_i c_i^2 = \frac{1}{3} \quad \text{e} \quad \sum_{i=1}^2 b_i c_i = \frac{1}{2} , \quad \text{calculamos:}$$

$$Q_f = \frac{1}{18} - \frac{1}{12} q(d+1) \quad , \quad \text{portanto}$$

$$Q_f^{(k)} = \frac{1}{18} - \frac{1}{12} q_k(d_k+1) \quad .$$

Como a condição de ordem para a árvore $f)$ é dada por :

$$\sum_{k=1}^2 \rho_k Q_f^{(k)} = \frac{1}{60} \quad , \quad \text{então usando (1.7b) e (1.9) , obtemos :}$$

$$\sum_{k=1}^2 \rho_k d_k q_k = -\frac{1}{30} \quad , \quad \text{que contraria a fórmula (1.13) . Portanto ordem}$$

cinco não é atingível .

■

- Análise de Estabilidade .

O conhecido método de Hammer-Hollingsworth [Butcher,87] , utiliza um processador , tem dois estágios e ordem quatro . Ele é A-estável , se baseia em quadratura de Gauss-Legendre [Butcher,87] e é dado por :

$-\frac{1}{2} - \frac{1}{6}\sqrt{3}$	$-\frac{1}{4}$	$-\frac{1}{4} - \frac{1}{6}\sqrt{3}$
$-\frac{1}{2} + \frac{1}{6}\sqrt{3}$	$-\frac{1}{4} + \frac{1}{6}\sqrt{3}$	$-\frac{1}{4}$
	$-\frac{1}{2}$	$-\frac{1}{2}$

Nosso objetivo será obter um método 4-RK , L-estável , do Tipo I , com implementação (2,1,2) , porque dessa forma teremos um método mais eficiente que o citado acima .

A função de estabilidade do método combinado (1.7a) é dada por

$$R(z) = \sum_{k=1}^2 \rho_k R_k^*(z) \quad , \quad \text{onde } R_k^*(z) \text{ é a função de estabilidade do}$$

k -ésimo método interno . Para não sobrecarregar a notação omitiremos o índice k e quando for conveniente voltaremos a usá-lo .

Nós supusemos os métodos internos possuidores de ordem três , então lembrando da fórmula (0.4.1) e tendo [Butcher,87] como referência , podemos escrever :

$$R^*(z) = \frac{T(z)}{\det(I - zA)} = \exp(z) + O(z^4) .$$

Como $\det(I - zA) = 1 - qz + z^2(a_1a_2 - [c_1c_2 - a_2c_1 - a_1c_2 + a_1a_2]) =$
 $= 1 - qz + \frac{1}{2} (q - \frac{1}{3})z^2$, então

$$R^*(z) = [1 + (1 - q)z + \frac{1}{2} (\frac{2}{3} - q)z^2] / [1 - qz + \frac{1}{2} (q - \frac{1}{3})z^2] .$$

A função $R^*(z)$ é analítica em $\mathbb{C}^-$, se os seus polos , dados por :

$$z = \frac{q \pm \sqrt{q^2 - 2q + 2/3}}{(q - 1/3)} ,$$

tiverem parte real positiva . Isto é verdade se $q > 1/3$.

$R^*(z)$ é A-estável se e somente se $|R^*(z)| \leq 1$ para $\forall z \in \mathbb{C}^-$, porém , pelo princípio do módulo máximo , basta analisarmos se $|R^*(it)| \leq 1$, $t \in \mathbb{R}$ (fronteira de $\mathbb{C}^-$) ; isto ocorre se e somente se $q \geq 1/2$. Como estamos interessados em L-estabilidade , precisamos satisfazer mais uma condição , dada por :

$$\lim_{z \rightarrow \infty} R(z) = 0 ,$$

ou equivalentemente ,

$$\lim_{z \rightarrow \infty} (\rho_1 R_1^*(z) + \rho_2 R_2^*(z)) = 0 \Leftrightarrow$$

$$\rho_1 (\frac{2}{3} - q_1) / (q_1 - \frac{1}{3}) + \rho_2 (\frac{2}{3} - q_2) / (q_2 - \frac{1}{3}) = 0 \quad (1.14)$$

Lembrando que as condições de ordem quatro , para o método combinado (1.7a) , requerem a solução de :

$$\begin{aligned}\rho_1 + \rho_2 &= 1 \\ q_1 \rho_1 + q_2 \rho_2 &= \frac{1}{2},\end{aligned}$$

e considerando $q_1 \neq q_2$, obtemos

$$\rho_1 = (q_2 - \frac{1}{2}) / (q_2 - q_1), \quad \rho_2 = - (q_1 - \frac{1}{2}) / (q_2 - q_1).$$

Substituindo estes valores em (1.14), obtemos a condição necessária para ocorrer L-estabilidade:

$$q_1 q_2 - \frac{2}{3} (q_1 + q_2) + \frac{7}{8} = 0. \quad (1.15)$$

Notemos que $q_1 \neq q_2$ é justificado, pois se $q_1 = q_2$, então através das condições de ordem quatro obteríamos $q_1 = q_2 = \frac{1}{2}$; portanto o limite de $R(z)$ seria 1, contrariando nossa condição inicial.

Nós agora colocamos $\kappa = q_1 + q_2$, e por (1.15), obtemos

$$q_1 q_2 = \frac{2}{3} \kappa - \frac{7}{8}.$$

Agora podemos escrever $R = \frac{P}{Q}$, onde

$$P(z) = 1 + (1 - \kappa)z + (\frac{1}{6} \kappa - \frac{2}{9})z^2 + \frac{1}{6} (\kappa - 1)z^3$$

$$Q(z) = 1 - \kappa z + (\frac{7}{6} \kappa - \frac{13}{18})z^2 + (\frac{7}{18} - \frac{1}{2} \kappa)z^3 + (\frac{1}{12} \kappa - \frac{5}{72})z^4.$$

A função R é analítica em $\mathbb{C}^-$ se $q_1, q_2 > \frac{1}{3}$, porque cada uma das funções R^* é analítica em $\mathbb{C}^-$ com $q_k > \frac{1}{3}$. Agora, vamos verificar se a condição $|R(i\omega)| \leq 1$.

$$|P(i\omega)/Q(i\omega)| = |R(i\omega)| \leq 1 \Leftrightarrow |Q(i\omega)|^2 - |P(i\omega)|^2 \geq 0 \Leftrightarrow$$

$$\frac{1}{36} (1 - \kappa)(\frac{5}{6} - \kappa)\omega^6 + \frac{1}{144} (\frac{5}{6} - \kappa)\omega^8 \geq 0 \Leftrightarrow \kappa \leq \frac{5}{6} \text{ ou } \kappa \geq 1.$$

Sujeito à $q_1, q_2 > \frac{1}{3}$ e $q_1 \neq \frac{2}{3}$, achamos, por (1.15),

$$q_2 = (\frac{2}{3} q_1 - \frac{7}{18}) / (q_1 - \frac{2}{3}),$$

e assim nós obtemos $\kappa = (q_2^2 - \frac{7}{18}) / (q_1 - \frac{2}{3})$; logo

$$\kappa \leq \frac{5}{6} \iff (q_1^2 - \frac{5}{6} q_1 + \frac{1}{6}) / (q_1 - \frac{2}{3}) \leq 0 \quad e$$

$$\kappa \geq 1 \iff (q_1^2 - q_1 + \frac{5}{18}) / (q_1 - \frac{2}{3}) \geq 0 \quad .$$

A primeira equivalência ocorre se e somente se $q_1 \leq \frac{1}{3}$ ou $\frac{1}{2} \leq q_1 \leq \frac{2}{3}$, porém $q_1 < \frac{2}{3}$ implica $q_2 \leq \frac{1}{3}$; como estamos interessados em $q_1, q_2 > \frac{1}{3}$, esta equivalência é descartada . A segunda equivalência ocorre se e somente se $q_1 > \frac{2}{3}$, então esta equivalência é consistente com L-estabilidade .

Juntando todas as condições , formalizamos os requisitos básicos para obter um método 4-RK , ordem quatro , 2 processadores , L-estável : (Veja as observações posteriores)

$$a) \quad q_1 \in [\frac{5}{6} , \frac{2}{3} + \frac{1}{6}\sqrt{2}) \quad , \quad q_2 \in (\frac{2}{3} + \frac{1}{6}\sqrt{2} , 1) \quad ,$$

$$b) \quad \rho_1 = (q_2 - \frac{1}{2}) / (q_2 - q_1) \quad , \quad \rho_2 = - (q_1 - \frac{1}{2}) / (q_2 - q_1) \quad ,$$

$$c) \quad \rho_1 d_1 + \rho_2 d_2 = 0 \quad , \quad |d_1| \quad , \quad |d_2| \leq \frac{1}{3} \quad ,$$

e para $k = 1, 2$:

$$c_1^{(k)} = \frac{1}{2} [d_k + 1 - \sqrt{(d_k^2 + \frac{1}{3})}] \quad ,$$

$$d) \quad c_2^{(k)} = \frac{1}{2} [d_k + 1 + \sqrt{(d_k^2 + \frac{1}{3})}] \quad ,$$

$$e) \quad a_1^{(k)} = \frac{1}{2} [q_k + d_k(1 - q_k) / \sqrt{(d_k^2 + \frac{1}{3})}] \quad ,$$

$$a_2^{(k)} = \frac{1}{2} [q_k - d_k(1 - q_k) / \sqrt{(d_k^2 + \frac{1}{3})}] \quad ,$$

$$f) \quad b_1^{(k)} = (c_2^{(k)} - \frac{1}{2}) / (c_2^{(k)} - c_1^{(k)}) \quad ,$$

$$b_2^{(k)} = - (c_1^{(k)} - \frac{1}{2}) / (c_2^{(k)} - c_1^{(k)}) \quad .$$

Observações :

1) as expressões obtidas no item d) foram calculadas através das fórmulas (1.8a) e $c_1 + c_2 - 1 = d$.

2) as expressões obtidas no item e) foram calculadas através da fórmula (1.11).

3) no item c) , $|d_k| \leq \frac{1}{3}$ porque queremos $c_i^{(k)} \in [0,1]$,
 $i = 1,2$, $k = 1,2$.

4) quando $q_1 = q_2$ obtemos , da equação $q_2 = (\frac{2}{3} q_1 - \frac{7}{18}) / (q_1 - \frac{2}{3})$,
 o valor $q_1 = q_2 = \frac{2}{3} + \frac{1}{6}\sqrt{2}$; quando $q_2 = 1$, nesta mesma equação ,
 obtemos $q_1 = \frac{5}{6}$. Por estes motivos , escolhemos em a) aqueles
 intervalos para q_1 e q_2 .

Exemplo 5 : Método do Tipo I , L-estável , com $q_1 = \frac{5}{6}$ e $d_1 = \frac{2}{9}$:

$$\begin{array}{c|cc}
 \frac{11}{18} - \frac{\sqrt{31}}{18} & \frac{5}{12} + \frac{\sqrt{31}}{186} & \frac{7}{36} - \frac{17\sqrt{31}}{279} \\
 \frac{11}{18} + \frac{\sqrt{31}}{18} & \frac{7}{36} + \frac{17\sqrt{31}}{279} & \frac{5}{12} - \frac{\sqrt{31}}{186} \\
 \hline
 3 & \frac{1}{2} + \frac{\sqrt{31}}{31} & \frac{1}{2} - \frac{\sqrt{31}}{31}
 \end{array} + \begin{array}{c|cc}
 \frac{1}{3} & \frac{1}{2} & -\frac{1}{6} \\
 1 & \frac{1}{2} & \frac{1}{2} \\
 \hline
 -2 & \frac{3}{4} & \frac{1}{4}
 \end{array}$$

Exemplo 6 : Método do Tipo I , L-estável , com $q_1 = \frac{5}{6}$, $d_1 = 0$,
 este método possui o mesmo vetor c do método de Hammer-Hollingsworth
 em ambos os métodos internos :

$$\begin{array}{c|cc}
 \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{5}{12} & \frac{1}{12} - \frac{\sqrt{3}}{6} \\
 \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{12} + \frac{\sqrt{3}}{6} & \frac{5}{12} \\
 \hline
 3 & \frac{1}{2} & \frac{1}{2}
 \end{array} + \begin{array}{c|cc}
 \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{2} & -\frac{\sqrt{3}}{6} \\
 \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{\sqrt{3}}{6} & \frac{1}{2} \\
 \hline
 -2 & \frac{1}{2} & \frac{1}{2}
 \end{array}$$

Exemplo 7 : Vamos considerar o seguinte PVI : ([Lambert,73] , p. 128)

$$\begin{aligned} y' &= \exp(10(t - y)) , \\ y(0) &= 0.1 , \quad 0 \leq t \leq 1 . \end{aligned}$$

A solução exata desse problema é dada por :

$$y(t) = 10^{-1} \ln(\exp(10t) + \exp(1) - 1) .$$

Resolvemos este problema numericamente , utilizando o método do exemplo 6 . Para $t = 1$ a solução exata nos fornece $y(1) = 1.000007801$. Os resultados numéricos forneceram a seguinte tabela :

h (tamanho do passo)	y_n (aproximação de $y(t_n)$, $t_n = 1$)
0.5	1.000320589
0.05	1.0000077679
0.005	1.0000078007

A implementação desse método encontra-se no apêndice , programa "Método do Tipo I" .

Observações :

1) Não podemos obter estabilidade algébrica para os métodos do Tipo I , porque isto exige que o vetor b tenha suas componentes positivas e esta hipótese , juntamente com o fato dos vetores $b^{(k)}$ possuírem componentes positivas (devido à condição f)) , ocasionariam uma contradição , porque pela igualdade

$$\rho_k b_i^{(k)} = b_{2(k-1)+i} ,$$

apresentada na observação 1) da subseção 1.4.1 , teríamos que $\rho_k \geq 0$. Isto não pode ocorrer , porque , pelas nossas condições , $q_2 > q_1 \geq \frac{5}{6}$ e isto implica $\rho_2 < 0$.

2) Implementação (2,1,2) para métodos do Tipo I fornece métodos embutidos , no método original , que são A-estáveis , possuem três estágios e ordem três . Estes métodos são aqueles que nós chamamos de internos , e eles servem para monitorar o erro do método combinado .

1.4.2 - Métodos do Tipo II .

Vamos considerar métodos do Tipo II , de quatro estágios , com implementação $\langle 2,2,1 \rangle$. Nosso objetivo é obter parâmetros que nos forneçam um método de ordem quatro que seja L-estável . As condições de ordem novamente serão obtidas através da Teoria das Diferenciais Elementares . Lembramos também que a matriz genérica de um método do Tipo II , com as características acima , foi apresentada no comentário inicial da seção 1.4 .

- Análise de Ordem .

Como estamos interessados em ordem quatro , vamos precisar das equações de ordem referentes às árvores de até quatro vértices . Estas informações estão no capítulo 0 , na tabela do exemplo 10 . Utilizando os coeficientes do método do Tipo II , dados em (1.6) , a tabela citada se transforma na tabela 1.3 abaixo :

Tabela 1.3 : Árvores de até quatro vértices e suas equações de ordem .

Árvore	Equação de Ordem
a)	$b_1 + b_2 + b_3 + b_4 = 1$
b)	$b_1 c_1 + b_2 c_2 + b_3 c_3 + b_4 c_4 = \frac{1}{2}$
c)	$b_1 c_1^2 + b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \frac{1}{3}$
d)	$b_1 c_1^2 + b_2 c_2^2 + b_3 [(c_3^2 - (c_2 - c_1)d_1 - (c_3 - c_1)(c_3 - c_2))] +$ $+ b_4 [c_4^2 - (c_4 - c_1)(c_4 - c_2) + (c_2 - c_1)d_2] = \frac{1}{6}$
e)	$b_1 c_1^3 + b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \frac{1}{4}$
f)	$b_1 c_1^3 + b_2 c_2^3 + b_3 c_3 [c_3^2 - (c_2 - c_1)d_1 - (c_3 - c_1)(c_3 - c_2)] +$ $+ b_4 c_4 [c_4^2 + (c_2 - c_1)d_2 - (c_4 - c_1)(c_4 - c_2)] = \frac{1}{8}$

g)

$$b_1 c_1^3 + b_2 c_2^3 + b_3 [c_3^3 - (c_2^2 - c_1^2) d_1 - (c_3 - c_1)(c_3^2 - c_2^2)] + \\ + b_4 [c_4^3 + (c_2^2 - c_1^2) d_2 - (c_4^2 - c_1^2)(c_4 - c_2)] = \frac{1}{12}$$

h)

$$\left\{ \begin{array}{l} b_1 c_1^3 + b_2 c_2^3 + \\ b_3 [c_3^3 - (c_1 + c_2 + c_3)(c_3 - c_1)(c_3 - c_2) - (c_2 - c_1)(2c_1 + c_2)d_1] + \\ b_4 [c_4^3 - (c_1 + c_2 + c_4)(c_4 - c_1)(c_4 - c_2) + (c_2 - c_1)(2c_2 + c_1)d_2] = \\ = \frac{1}{24} \end{array} \right.$$

Definimos

$$\bar{d}_1 = (c_2 - c_1)b_3 d_1, \quad \bar{d}_2 = - (c_2 - c_1)b_4 d_2;$$

$$\bar{b}_3 = (c_3 - c_1)(c_3 - c_2)b_3, \quad \bar{b}_4 = (c_4 - c_1)(c_4 - c_2)b_4.$$

Resolvendo o sistema linear formado pelas equações a) , b) , c) e e) da tabela 1.3 , obtemos :

$$\bar{b}_3 = \frac{- \left[\frac{1}{4} - \frac{1}{3} (c_1 + c_2 + c_4) + \frac{1}{2} (c_1 c_2 + c_1 c_4 + c_2 c_4) - c_1 c_2 c_4 \right]}{(c_4 - c_3)} \quad (1.16)$$

$$\bar{b}_4 = \frac{\left[\frac{1}{4} - \frac{1}{3} (c_1 + c_2 + c_3) + \frac{1}{2} (c_1 c_2 + c_1 c_3 + c_2 c_3) - c_1 c_2 c_3 \right]}{(c_4 - c_3)} \quad (1.17)$$

Agora , substituindo a equação c) na equação d) , dadas na tabela 1.3 , obtemos

$$\bar{d}_1 + \bar{d}_2 = \frac{1}{6} - \bar{b}_3 - \bar{b}_4,$$

e substituindo a equação e) na equação f) obtemos

$$c_3 \bar{d}_1 + c_4 \bar{d}_2 = \frac{1}{8} - c_3 \bar{b}_3 - c_4 \bar{b}_4.$$

Resolvendo o sistema linear em $\bar{d}_1$ e $\bar{d}_2$, obtemos

$$\begin{aligned}\bar{d}_1 &= \left(\frac{1}{6} c_4 - \frac{1}{8}\right) / (c_4 - c_3) - \bar{b}_3, \\ \bar{d}_2 &= -\left(\frac{1}{6} c_3 - \frac{1}{8}\right) / (c_4 - c_3) - \bar{b}_4.\end{aligned}\quad (1.18)$$

Substituindo a equação e) nas equações g) e h) obtemos respectivamente :

$$\begin{aligned}(c_1 + c_2)(\bar{d}_1 + \bar{d}_2) + (c_2 + c_3)\bar{b}_3 + (c_1 + c_4)\bar{b}_4 &= \frac{1}{6}, \\ (2c_1 + c_2)\bar{d}_1 + (2c_2 + c_1)\bar{d}_2 + (c_1 + c_2 + c_3)\bar{b}_3 + (c_1 + c_2 + c_4)\bar{b}_4 &= \frac{5}{24}.\end{aligned}$$

Substituindo os valores de (1.18) nas duas equações acima obtemos respectivamente :

$$(c_3 - c_1)\bar{b}_3 + (c_4 - c_2)\bar{b}_4 = \frac{1}{6} (1 - c_1 - c_2), \quad (1.19)$$

$$\begin{aligned}(c_3 - c_1)\bar{b}_3 + (c_4 - c_2)\bar{b}_4 &= \frac{5}{24} - \frac{1}{6} [(c_1 + c_2) + (c_1 c_4 - c_2 c_3) / (c_4 - c_3)] + \\ &\quad - \frac{1}{18} [(c_2 - c_1) / (c_4 - c_3)].\end{aligned}$$

Igualando-se as duas equações em (1.19) nós obtemos :

$$(1 - 4c_1)c_4 - (1 - 4c_2)c_3 = 3(c_2 - c_1). \quad (1.20)$$

Substituindo os valores de $\bar{b}_3$ e $\bar{b}_4$ na fórmula (1.19), obtemos

$$\begin{aligned}& - (1 - 2c_1 - 6c_2 + 12c_1 c_2 + 6c_2^2 - 12c_1 c_2^2)c_3 + \\ & \quad + (1 - 6c_1 - 2c_2 + 6c_1^2 + 12c_1 c_2 - 12c_1^2 c_2)c_4 = \\ & = 3(c_2 - c_1) - 4(c_2^2 - c_1^2) + 6c_1 c_2 (c_2 - c_1).\end{aligned}\quad (1.21)$$

As equações (1.20) e (1.21) são lineares em c_3 e c_4 e o determinante associado a este sistema assume o seguinte valor :

$$2(c_2 - c_1)(c_1 + c_2 - 6c_1 c_2).$$

Então, sujeito a $c_1 \neq c_2$ e $c_1 + c_2 \neq 6c_1 c_2$, o sistema formado pelas equações (1.20) e (1.21) possui solução dada por :

$$c_3 = 1 - c_1, \quad c_4 = 1 - c_2.$$

Agora , substituindo os valores de c_3 e c_4 , encontrados acima , nas equações (1.16) e (1.17) obtemos

$$b_1 = b_3 = \frac{-(c_2^2 - c_2 + \frac{1}{6})}{2(c_2 - c_1)(1 - c_1 - c_2)} \quad , \quad (1.22)$$

$$b_2 = b_4 = \frac{(c_1^2 - c_1 + \frac{1}{6})}{2(c_2 - c_1)(1 - c_1 - c_2)} \quad . \quad (1.23)$$

Usando (1.18) , juntamente com estes resultados , podemos escrever :

$$d_1 = \frac{(1 - c_1 - c_2)(2c_1 - \frac{(1 - 2c_2)(1 - 6c_2)}{12(c_2^2 - c_2 + \frac{1}{6})})}{(c_2 - c_1)} \quad ,$$

$$d_2 = \frac{-(1 - c_1 - c_2)(2c_2 - \frac{(1 - 2c_1)(1 - 6c_1)}{12(c_1^2 - c_1 + \frac{1}{6})})}{(c_2 - c_1)}$$

Então , qualquer escolha de c_1 e c_2 tal que

$$c_1 \neq c_2 \quad , \quad c_1, c_2 \neq \frac{1}{2} \pm \frac{\sqrt{3}}{6} \quad \text{e} \quad c_1 + c_2 \neq 1 \quad ,$$

dá um método do Tipo II com ordem quatro .

- Análise de Estabilidade

Vamos estudar propriedades de estabilidade do método do Tipo II obtido acima . A função de estabilidade para tal método assume a forma

$$R(z) = 1 + \sum_{i=1}^4 b_i W_i, \quad \text{onde}$$

$$W_1 = \frac{z}{(1 - zc_1)}, \quad W_2 = \frac{z}{(1 - zc_2)},$$

$$W_3 = \frac{z^2 d_1}{(1 - zc_1)^2} + \frac{z^2 (c_3 - c_1 - d_1)}{(1 - zc_1)(1 - zc_2)} + \frac{z}{(1 - zc_1)},$$

$$W_4 = \frac{z^2 d_2}{(1 - zc_2)^2} + \frac{z^2 (c_4 - c_2 - d_2)}{(1 - zc_1)(1 - zc_2)} + \frac{z}{(1 - zc_2)}.$$

É interessante analisarmos o seguinte limite

$$\begin{aligned} R_\infty = \lim_{z \rightarrow \infty} R(z) = 1 - \frac{b_1}{c_1} - \frac{b_2}{c_2} + \frac{b_3 d_1}{c_1^2} + \frac{(c_3 - c_1 - d_1)b_3}{c_1 c_2} + \\ - \frac{b_3}{c_1} + \frac{b_4 d_2}{c_2^2} + \frac{b_4 (c_4 - c_2 - d_2)}{c_1 c_2} - \frac{b_4}{c_2}. \end{aligned}$$

Utilizando-se dos valores de b_1 e b_2 , expressos em (1.22) e (1.23) respectivamente, obtemos

$$R_\infty = 1 + \frac{\frac{1}{2} - c_1 - c_2}{c_1 c_2} + \frac{(c_2 - c_1)(c_2 d_1 b_3 - c_1 d_2 b_4)}{c_1^2 c_2^2},$$

substituindo os valores de b_3 e b_4 na expressão acima, obtemos

$$\begin{aligned} R_\infty = \frac{1}{c_1^2 c_2^2} \left[\frac{1}{24} - \frac{1}{3} (c_1 + c_2) + \frac{1}{2} (c_1^2 + 4c_1 c_2 + c_2^2) \right] + \\ - \frac{1}{c_1^2 c_2^2} [2c_1 c_2 (c_1 + c_2)] + 1. \end{aligned}$$

$$|R_\infty| \leq 1 \quad \text{se e somente se}$$

$$-2c_1^2 c_2^2 \leq \frac{1}{24} - (c_1 + c_2) \left[\frac{1}{3} - \frac{1}{2} (c_1 + c_2) + 2c_1 c_2 \right] + c_1 c_2 \leq 0 \Leftrightarrow$$

$$-2c_1^2 c_2^2 \leq \frac{-1}{2} (c_1 + c_2) \left[\frac{1}{6} - (c_1 + c_2) + 4c_1 c_2 \right] + \frac{1}{4} \left[4c_1 c_2 + \frac{1}{6} - (c_1 + c_2) \right] \leq 0 \Leftrightarrow$$

$$-2c_1^2 c_2^2 \leq \frac{1}{4} \left[\frac{1}{6} - (c_1 + c_2) + 4c_1 c_2 \right] [1 - 2(c_1 + c_2)] \leq 0 .$$

$$\text{Se } 4c_1 c_2 - (c_1 + c_2) + \frac{1}{6} \geq 0 \quad \text{e} \quad c_1 + c_2 \geq \frac{1}{2} , \quad \text{então}$$

$$|R_\infty| \leq 1 .$$

Segundo [Iserles-Norsett, 90] , p. 485 , pares de pontos (c_1, c_2) que permitem métodos A-estáveis e L-estáveis foram obtidos numericamente . Os resultados sugerem que A-estabilidade ocorre se e somente se os pares estão no conjunto :

$$\{(c_1, c_2) : 4c_1 c_2 - (c_1 + c_2) + \frac{1}{6} \geq 0 ; c_1, c_2 \in (\frac{1}{4}, 1] ; c_1 \neq c_2 ; c_1 + c_2 \neq 1 ; c_1, c_2 \neq \frac{1}{2} + \frac{\sqrt{3}}{6} \} , \quad (1.24)$$

e temos L-estabilidade se ainda ocorrer $R_\infty = 0$, ou seja , se

$$c_1^2 c_2^2 - 2c_1 c_2 (c_1 + c_2) + \frac{1}{2} (c_1^2 + 4c_1 c_2 + c_2^2) - \frac{1}{3} (c_1 + c_2) + \frac{1}{24} = 0 .$$

Exemplo 8 : Uma escolha de (c_1, c_2) que ocasiona um método L-estável é dada por $c_1 = \frac{1}{2}$ e $c_2 = \frac{2}{3}$, resultando no método :

$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{2}{3}$	0	$\frac{2}{3}$	0	0
$\frac{1}{2}$	$-\frac{5}{2}$	$\frac{5}{2}$	$\frac{1}{2}$	0
$\frac{1}{3}$	$-\frac{5}{3}$	$\frac{4}{3}$	0	$\frac{2}{3}$
	-1	$-\frac{3}{2}$	-1	$\frac{3}{2}$

Exemplo 9 : Seja novamente o PVI do exemplo 7 .

Utilizando o método dado no exemplo 8 obtemos os resultados exibidos na tabela abaixo :

h (tamanho do passo)	y_n (aproximação de $y(t_n)$, $t_n = 1$)
0.5	1.0008038822
0.05	1.0000077360
0.005	1.0000078007

A implementação deste método encontra-se no apêndice , programa "Método do Tipo II" .

Até o momento estudamos condições que permitem que métodos do Tipo II , com implementação $\langle 2,2,1 \rangle$, atinjam ordem quatro e sejam L-estáveis . Agora vamos investigar se tais métodos podem ser algebricamente estáveis . Para que isto ocorra , conforme vimos na definição 0.11 , as condições a seguir devem ser satisfeitas :

a) as componentes do vetor b devem ser positivas , ou seja , $b_i \geq 0$, $i = 1, \dots, 4$;

b) A matriz $M = BA + A^t B - bb^t$ deve ser semi-definida positiva , onde $B = \text{diag}(b_1, \dots, b_4)$.

Nós investigaremos a existência de pares (c_1, c_2) que estejam no conjunto (1.24) e satisfaçam à condição para L-estabilidade e ainda permitam que as condições a) , b) acima sejam satisfeitas .

Analisando as fórmulas (1.22) e (1.23) observamos que a condição a) vale se e somente se

$$\text{ou , } \quad \frac{1}{4} < c_1 < \frac{1}{2} + \frac{\sqrt{3}}{6} \quad \text{e} \quad \frac{1}{2} + \frac{\sqrt{3}}{6} < c_2 \leq 1 \quad ,$$

$$\text{ou , } \quad \frac{1}{4} < c_2 < \frac{1}{2} + \frac{\sqrt{3}}{6} \quad \text{e} \quad \frac{1}{2} + \frac{\sqrt{3}}{6} < c_1 \leq 1 \quad .$$

Para verificarmos se a condição b) é satisfeita , teremos que analisar os auto valores da matriz M , que indicaremos por λ_i , $i = 1, \dots, 4$. Inicialmente , vamos provar que λ_1 é nulo . Isto acontece porque M é singular . De fato :

$$M e = \begin{bmatrix} b_1(2c_1 + d_1 - 1) - b_2(2c_2 + d_2 - 1) \\ b_2(2c_2 + d_2 - 1) - b_1(2c_1 + d_1 - 1) \\ 0 \\ 0 \end{bmatrix}, \text{ onde } e = [1, \dots, 1]^t \in \mathbb{R}^4$$

Mas podemos verificar, por uma simples substituição de valores, que

$$b_1(2c_1 + d_1 - 1) = b_2(2c_2 + d_2 - 1), \quad (1.25)$$

logo $M e = 0$ (vetor nulo) e portanto M é singular.

O próximo passo é verificar que $\lambda_2 = 2b_1(2c_1 + d_1 + 2b_2 - 1)$ é um autovalor de M , que tem como um auto-vetor $v = [1, -1, 1, -1]^t$. De fato,

$$Mv = \begin{bmatrix} 2b_1c_1 - b_1d_1 - b_2(1 - 2c_2 - d_2) - 2b_1^2 + 2b_1b_2 \\ -2b_2c_2 - b_2d_2 + b_1(1 - 2c_1 - d_1) + 2b_2^2 - 2b_1b_2 \\ b_1(2d_1 + 3c_1 - 1) + b_1c_1 - 2b_1^2 + 2b_1b_2 \\ b_2(1 - 3c_2 - 2d_2) - b_2c_2 + 2b_2^2 - 2b_1b_2 \end{bmatrix},$$

logo, se usarmos (1.25), $Mv = 2b_1(2c_1 + d_1 + 2b_2 - 1)[1, -1, 1, -1]^t$.

Finalmente, nós procuramos valores de α para os quais os vetores $v_\alpha = [1, -\alpha, -1, -\alpha]^t$ sejam auto-vetores de M , associados aos autovalores λ_3 e λ_4 . Temos que

$$Mv = \begin{bmatrix} 2b_1c_1 - b_1d_1 + \alpha b_2(1 - 2c_2 - d_2) \\ \alpha(b_2d_2 - 2b_2c_2) - b_1(1 - 2c_1 - d_1) \\ b_1(d_1 - \alpha(1 - 2c_1 - d_1)) - 2b_1c_1 \\ b_2(1 - 2c_2 - d_2 - \alpha d_2) + 2\alpha b_2c_2 \end{bmatrix}.$$

Se usarmos a igualdade (1.25) teremos as equações:

$$b_1(2c_1 - d_1) + \alpha b_2(1 - 2c_2 - d_2) = \lambda(\alpha),$$

$$b_1(1 - 2c_1 - d_1) + \alpha b_2(2c_2 - d_2) = \alpha \lambda(\alpha).$$

As duas soluções da equação quadrática :

$$b_2(1 - 2c_2 - d_2)\alpha^2 + [b_1(2c_1 - d_1) - b_2(2c_2 - d_2)]\alpha - b_1(1 - 2c_1 - d_1) = 0 ,$$

permitem concluir que os v_α correspondentes , realmente são os auto-vetores associados aos auto-valores λ_3 e λ_4 .

Segundo [Iserles-Norsett,90] , p. 487 , experimentos computacionais sugerem que $\min \{ \lambda_1, \lambda_2, \lambda(\alpha) \} \leq 0$, se limitarmos a escolha dos pares (c_1, c_2) no quadrado $[0,1] \times [0,1]$. Dessa forma não existe uma escolha de um par (c_1, c_2) que permita que M seja considerada semi-definida positiva , logo o método em questão não é algebricamente estável .

1.5 Conclusão .

Vimos neste capítulo duas famílias de métodos de Runge-Kutta Paralelos . Estes métodos foram construídos utilizando a Teoria das Diferenciais Elementares . Obtemos métodos 4-RK implícitos , de ordem quatro , L-estáveis . Estes métodos são inovadores no estudo de métodos de Runge-Kutta , já que , se utilizarmos a teoria convencional , sob o prisma de implementação seqüencial , não conseguimos um método com estas propriedades .

Já sob o ponto de vista prático , observamos que os experimentos numéricos exibidos nos exemplos 7 e 9 estão de acordo com a teoria que diz respeito à ordem dos métodos .

CAPÍTULO 2

Métodos de Runge-Kutta Implícitos Iterados Diagonalmente por um Processo Paralelo..

2.0 - Introdução .

Neste capítulo vamos estudar uma técnica numérica para resolver problemas de valor inicial , desenvolvida no artigo :
[Houwen,Sommeijer,Couzy,89] . A idéia básica é usar um método de Runge-Kutta implícito para resolvermos numericamente o PVI , porém usaremos um método iterativo diagonal para resolver as equações implícitas do Runge-Kutta . Esta técnica facilita a implementação dos métodos implícitos e como veremos , os resultados numéricos obtidos são satisfatórios .

2.1 - Iteração Diagonal .

Iteração Diagonal é uma técnica iterativa que utilizaremos para resolver as equações implícitas de métodos de Runge-Kutta completamente implícitos . A vantagem de se usar tal técnica é que ela nos permite resolver estas equações implícitas , simultanea e independentemente umas das outras , em processadores que podem executar em paralelo .

Vamos introduzir agora algumas notações que serão utilizadas no decorrer do texto .

Seja o problema (0.1) . Vamos introduzir a seguinte função :

$F : \mathbb{R}^{sN} \rightarrow \mathbb{R}^{sN}$, tal que , se $Y = (Y_1, \dots, Y_s)^t \in \mathbb{R}^{sN}$, com $Y_i \in \mathbb{R}^N$ para $i = 1, \dots, s$, então $F(Y) = (f(Y_1), \dots, f(Y_s))^t$.

Definição 2.1 . O Produto Tensorial entre duas matrizes , $X = (x_{ij})$, com $i = 1, \dots, m_1$, $j = 1, \dots, n_1$ e $T = (t_{kl})$, com $k = 1, \dots, m_2$, $l = 1, \dots, n_2$, é uma matriz $W = (w_{i_1, i_2})$, com $i_1 = 1, \dots, m_1 m_2$, $i_2 = 1, \dots, n_1 n_2$, definida por :

$$W = X \otimes T = \begin{bmatrix} x_{11} T & \dots & x_{1n_1} T \\ \dots & \dots & \dots \\ x_{m_1 1} T & \dots & x_{m_1 n_1} T \end{bmatrix} .$$

Utilizando as definições acima , com suas respectivas notações , podemos escrever um método de Runge-Kutta de uma forma mais compacta :

$$Y = e \otimes y_{n-1} + h (A \otimes I_N) F(Y) , \quad (2.1)$$

$$y_n = y_{n-1} + h (b^t \otimes I_N) F(Y) , \quad (2.2)$$

onde $e = (1, \dots, 1)^t \in \mathbb{R}^s$ e I_N é a matriz identidade de ordem N .

Para resolvermos as equações (2.2) introduzimos o processo iterativo diagonal :

$Y^{(0)}$: aproximação inicial para o vetor Y (o método que fornece a aproximação inicial será chamado mais adiante de predictor)

$$Y^{(k)} - h (D \otimes I_N) F(Y^{(k)}) = e \otimes y_{n-1} + h [(A - D) \otimes I_N] F(Y^{(k-1)}) \quad (2.3)$$

$$y^{(k)} = y_{n-1} + h (b^t \otimes I_N) F(Y^{(k)}) , \quad k = 1, 2, \dots ,$$

onde D é uma matriz diagonal de dimensão s . É fácil notar que cada componente $Y_i^{(k)}$ da incógnita iterativa $Y^{(k)}$ é da forma :

$$Y_i^{(k)} - h d_i f(Y_i^{(k)}) - r_i = 0 , \quad i = 1, \dots, s , \quad (2.4)$$

onde r_i é uma constante . Como cada uma dessas componentes são independentes umas das outras , então podemos resolver as equações (2.4) em paralelo . Portanto , se tivermos s processadores disponíveis , cada uma das iterações em (2.3) requer a solução de um sistema não linear de dimensão N . Logo , se limitarmos o número de iterações , digamos a m iterações , então este processo iterativo tem o

Exemplo 1 : Dado um método de Runge-Kutta com matriz A e vetor de pesos b^t , podemos considerar o processo (2.3), com m iterações e com aproximação inicial $y^{(0)}$ e y_{n-1} , como um método de Runge-Kutta que possui representação matricial dada por : (nesta representação, 0 é uma matriz nula e 0^t é o vetor nulo)

2.1.1 - Ordem de Precisão .

$$y_n - y^{(k)} = h (b^t \otimes I_N) (F(Y) - F(Y^{(k)})) \quad (2.5)$$
$$FCY) - FCY^{(k)}) = h J_F [(A - D) \otimes I_N](FCY) - FCY^{(k-1)}) + \\ + h J_F [D \otimes I_N](FCY) - FCY^{(k)}) , \quad (2.6)$$

onde $J_F = J_F(e \otimes y_{n-1}) = \text{diag}(J_f)$ é a matriz jacobiana, de ordem N_s , da função F , e J_f é a matriz jacobiana, de ordem N , da função f , calculada no ponto y_{n-1} . Podemos ainda escrever o lado esquerdo de (2.6) como:

$$h[I_{N_s} - h J_F (D \otimes I_N)]^{-1} J_F [(A - D) \otimes I_N] (F(Y) - F(Y^{(k-1)})) ,$$

onde $I_{N_s} = \text{diag}(I_N)$. Usando recursão, finalmente temos:

$$F(Y) - F(Y^{(k)}) = Z^k (F(Y) - F(Y^{(0)})) , \quad k = 1, 2, \dots ,$$

onde Z é uma matriz de recursão definida por

$$Z = h[I_{N_s} - h J_F (D \otimes I_N)]^{-1} J_F [(A - D) \otimes I_N] ,$$

e $Y^{(0)}$ é uma aproximação inicial para o vetor Y .

Agora, podemos escrever (2.5) como

$$y_n - y^{(k)} = h(b^t \otimes I_N) Z^k (F(Y) - F(Y^{(0)})) , \quad k = 1, 2, \dots .$$

Se $Y - Y^{(0)} = O(h^q)$, então $y_n - y^{(k)} = O(h^{q+k+1})$, $k = 1, 2, \dots$. Este resultado será usado quando formos estudar, na próxima seção, ordem de métodos PDIRK.

2.2 - Métodos PDIRK (RK Paralelo, Iterado Diagonalmente).

Os métodos PDIRK são construídos limitando-se o número de iterações no processo (2.3). Pelo exemplo 1 da seção 2.1 observamos que estes métodos pertencem à classe dos métodos de Runge-Kutta Diagonalmente Implícitos (DIRK) [Norsett, 74], ou seja, as matrizes correspondentes a estes métodos são triangulares inferiores. No capítulo 1 nós estudamos uma família de métodos que são diagonalmente implícitos, e que nós chamamos de métodos do Tipo II, vimos que estes métodos tinham potencial para paralelismo. Também vamos explorar potencial para paralelismo de métodos PDIRK.

2.2.1 - Ordem de Métodos PDIRK .

Para simplificar as notações , assumiremos que a equação $y'(t) = f(y(t))$ é escalar . A extensão para sistemas de EDO não é difícil , basta utilizar a notação de produto tensorial dada na seção 2.1 . Nós vamos aproveitar os cálculos feitos na subseção 2.1.1 , para analisar a ordem de precisão de um método PDIRK . Então , se y_n representa a aproximação para $y(t_n)$, quando utilizamos um método s-RK e $y^{(m)}$ representa a solução final do processo iterativo (2.3) , podemos escrever :

$$y_n - y^{(m)} = O(h^{q+m+1}) , \text{ se } Y - Y^{(0)} = O(h^q) .$$

Neste caso dizemos que $y^{(m)}$ aproxima y_n com ordem global $q + m$. O método que produz a aproximação inicial $Y^{(0)}$ para a solução de $Y = y_{n-1}e + hAF(Y)$ será chamado de previsor . Nesta subseção usaremos métodos previsores da forma :

$$Y^{(0)} = y_{n-1}e + h E F(y_{n-1}e) + h B F(Y^{(0)}) \quad (2.7)$$

A matriz B será diagonal , porque estamos interessados em explorar paralelismo , enquanto que a matriz E será arbitrária .

Podemos reescrever as equações de Y e $Y^{(0)}$ da seguinte forma :

$$Y^{(0)} = y_{n-1}e + h E F(y_{n-1}e) + h B [F(y_{n-1}e) + h J_r (E F(y_{n-1}e) + B F(y_{n-1}e) + O(h))] ,$$

$$Y = y_{n-1}e + h A [F(y_{n-1}e) + h J_r A (F(y_{n-1}e) + O(h))] .$$

É fácil notar que o previsor (2.7) possui sempre ordem 1 em h e atinge ordem dois se $[A - (E + B)]F(y_{n-1}e) = 0$ (ou equivalentemente , se $[A - (E + B)]e = 0$) , e atinge ordem três se , além da hipótese anterior , também ocorrer : $[A J_r A - B J_r A](F(y_{n-1}e)) = 0$ (ou equivalentemente , se $[A^2 - BA]e = 0$.

Observações :

1) Lembremos que no caso escalar a matriz jacobiana Jf é a derivada de f calculada no ponto y_{n-1} , ou seja, $f'(y_{n-1})$; lembramos também que $Jf = \text{diag}(Jf)$;

2) O método de Runge-Kutta (0.2) tem ordem global p quando $y_n - y = O(h^{p+1})$.

Teorema 2.1 . Seja um método de Runge-Kutta com s estágios e ordem global p^* . Então a aproximação $y^{(m)}$, gerada pelo método PDIRK, tem ordem global $p = \min \{p^*, m + q\}$, onde q assume respectivamente os valores 1, 2 ou 3 se ocorrerem os respectivos casos :

- 1) as matrizes B e E são arbitrárias ;
- 2) $(E + B)e = Ae$;
- 3) vale o caso 2) e $(B - A)e = A^2e$.

Prova : Basta observar que $y_n - y^{(m)} = O(h^{q+m+1})$, quando o predictor é de ordem q em h ; como por hipótese, $y_n - y = O(h^{p^*+1})$, segue que $y^{(m)} - y = O(h^{q+m+1}) + O(h^{p^*+1})$, então conforme os casos 1), 2) ou 3), q assume respectivamente os valores 1, 2 ou 3, devido ao comentário anterior ao teorema, e dessa forma segue o resultado.

■

Agora daremos exemplos de matrizes B e E , associadas ao teorema acima, as quais nos fornecerão alguns tipos de métodos PDIRK.

Exemplo 2 : $E = B = 0$ (matriz nula) .

Então, por (2.7), $y^{(0)} = y_{n-1}e$. A ordem do método PDIRK obtido, depois de m iterações, é igual a $p = \min \{p^*, m + 1\}$. A representação matricial desse método já foi dada no exemplo 1), seção 2.1. Lembremos que a matriz D é não nula, porque queremos métodos diagonalmente implícitos. Denotaremos este método por método do Tipo A.

Exemple 3 : $E = 0$, $B = D$, $e Y^{(0)} = y_{n-1} e + h D F(Y^{(0)})$.

Depois de executadas m iterações, a ordem desse método PDIRK é $p = \min \{p^*, m + q\}$, onde q assume os valores 1 ou 2, conforme ocorrerem os respectivos casos:

- 1) $D \neq 0$,
- 2) $D = \text{diag } (A\theta)$.

A representação matricial desse método é dada por :

[illegible]

Denotaremos este método por método do Tipo B.

Pretendemos implementar o método do Tipo B.1 (Tipo B , caso 1) , o qual possui matriz diagonal D de elementos iguais . O valor do elemento diagonal da matriz D será dado posteriormente na Tabela 2.2 .

Exemple 4 : $B = D$, $Y^{(0)} = y_{n-1} + h E F(y_{n-1}, e) + h D F(Y^{(0)})$.

Depois de executadas m iterações, a ordem desse método PDIRK é $p = \min \{p^*, m + q\}$, onde q assume os valores 1, 2 ou 3, conforme ocorrerem os respectivos casos:

- 1) $D \neq 0, E \neq 0$;
- 2) $D = \text{diag}(Ae - Ee), E \neq 0$;
- 3) $D = \text{diag}(A - E)e, DAe = A^2e$.

Denotaremos este método por método do Tipo C . A sua representação matricial é dada por :

$$y_n - y^{(m)} = e_0^t [hCA - D][FCY - FCY^{(m-1)}] + h D [FCY - FCY^{(m)}] ,$$

Usando os resultados obtidos na subseção 2.1.1 concluímos que

$$y_n - y^{(m)} = h e_0^t [(A - D) + DZ]Z^{m-1}[FCY - FCY^{(0)}] = O(h^{m+q}) ,$$

onde q é a ordem em h do previsor . Assim , com este comentário , podemos enunciar o teorema abaixo , sem necessidade de demonstração , já que ele é similar ao teorema 2.1 .

Teorema 2.2 . Seja um método de Runge-Kutta "stiffly" preciso , com s estágios e com ordem global p^* . Então a aproximação $y^{(m)} = e_0^t Y^{(m)}$, gerada pelo método PDIRK , tem ordem $p = \min \{p^*, m + q - 1\}$, onde q assume os valores 1 , 2 ou 3 , conforme ocorrerem respectivamente os casos :

- 1) as matrizes B e E são arbitrárias ;
- 2) $(E + B)e = Ae$;
- 3) vale o caso 2) e $BAe = A^2e$.

■

Observações :

1) Os mesmos métodos obtidos nos exemplos 2) , 3) e 4) podem ser considerados como métodos PDIRK com aproximação $y^{(m)} = e_0^t Y^{(m)}$, porém as ordens desses métodos , segundo o teorema 2.2 , são dadas por $p = \min \{p^*, m + q - 1\}$; além disso , o vetor b é igual a $e_0^t A$.

2) Os métodos do Tipo B.2 (exemplo 3 , caso 2) são completamente determinados pelo método s-RK , enquanto que os métodos do Tipo C.3 (exemplo 4 , caso 3) fornecem a matriz D mas dependem de informações a respeito da soma das linhas da matriz E .

3) Diremos que um método PDIRK requer n estágios sequenciais , se em cada passo forem resolvidas n relações implícitas . Então , métodos do Tipo A requerem m estágios sequenciais , ao passo que os métodos dos Tipos B e C requerem $m + 1$ estágios sequenciais .

2.3 - Métodos PDIRK de Classes I e II .

Passaremos a dividir os métodos PDIRK em duas classes :

- classe I : métodos PDIRK que possuem aproximação $y^{(m)}$ dada pelo processo iterativo (2.3) (escalar) ;
- classe II : métodos PDIRK que possuem aproximação $y^{(m)}$ dada pela fórmula (2.8) .

2.3.1 - Estabilidade de Métodos das Classes I e II .

O nosso objetivo é obter uma fórmula de recorrência que produza expressões para as funções de estabilidade dos métodos das classes I e II , que indicaremos respectivamente por $R_m^1(z)$ e por $R_m^{11}(z)$. Usaremos as notações y_n^1 e y_n^{11} para indicar as aproximações $y^{(m)}$ dadas respectivamente por (2.3) (escalar) e (2.8) . As funções de estabilidade são determinadas por :

$$y_n^1 = y_{n-1} + z b^t Y^{(m)} = R_m^1(z) y_{n-1} ,$$

$$y_n^{11} = e^t Y^{(m)} = R_m^{11}(z) y_{n-1} .$$

Usando o problema teste escalar : $y'(t) = \lambda y(t)$, temos :

$$Y^{(0)} = y_{n-1} e + zE y_{n-1} e + zB Y^{(0)} \Rightarrow (I - zB) Y^{(0)} = (I + zE) y_{n-1} e \Rightarrow$$

$$\Rightarrow Y^{(0)} = (I - zB)^{-1} (I + zE) y_{n-1} e$$

e também temos :

$$Y^{(k)} = y_{n-1} e + z(A - D) Y^{(k-1)} + zD Y^{(k)} \Rightarrow$$

$$\Rightarrow Y^{(k)} = (I - zD)^{-1} (y_{n-1} e + z(A - D) Y^{(k-1)}) =$$

$$= (I - zD)^{-1} (y_{n-1} e + z(A - D)(I - zD)^{-1} Q_{k-1} y_{n-1} e) ,$$

onde Q_k é definida pela seguinte fórmula de recorrência :

$$Q_0 = (I - zD)(I - zB)^{-1}(I + zE) ,$$

$$Q_k = I + zQ_{k-1} , \quad Z = Z(z) = z(A - D)(I - zD)^{-1} .$$

Observação : Para provar a fórmula de recorrência anterior , basta usar indução e observar que a matriz Q_{k-1} aparece na expressão de $Y^{(k)}$.

Finalmente , podemos escrever

$$Y^{(k)} = (I - zD)^{-1} Q_k y_{n-1} e ,$$

logo

$$Y^{(m)} = (I - zD)^{-1} Q_m y_{n-1} e .$$

Portanto ,

$$R_m^1(z) = 1 + zb^t(I - zD)^{-1} Q_m e ,$$

$$R_m^{11}(z) = e_e^t (I - zD)^{-1} Q_m e . \quad (2.9)$$

Podemos achar Q_m pela fórmula

$$\begin{aligned} Q_m &= Q_m(z) = I + Z + Z^2 + \dots + Z^{m-1} + Z^m Q_0 = \\ &= I + Z + Z^2 + \dots + Z^{m-1} + Z^m(I - zD)(I - zB)^{-1}(I + zE) . \end{aligned}$$

Nós vamos considerar o estudo de estabilidade em dois casos separadamente : primeiro vamos supor que ambas as matrizes B e D tenham em suas diagonais elementos iguais ; depois vamos assumir que ambas as matrizes B e D tenham elementos diagonais arbitrários .

2.3.2 - Métodos com Matriz Diagonal de Elementos Iguais .

Queremos obter um parâmetro para construir métodos das classes I e II que sejam A-estáveis ou L-estáveis . Neste caso , vamos procurar um número d tal que a matriz $D = dI$, da iteração diagonal , produza estas características de estabilidade .

Seja $B = bI$, e $D = dI$; então a matriz $Q_m(z)$, obtida no final da subseção anterior , pode ser reescrita como :

$$Q_m = \frac{N_m(z)}{(1 - bz)(1 - zd)^{m-1}} ,$$

onde $N_m(z)$ é um polinômio em z que tem matrizes como coeficientes . Agora , podemos escrever as funções de estabilidade como :

$$R_m^I(z) = 1 + \frac{b^t z N_m(z) e}{(1 - zb)(1 - zd)^m} , \quad (2.10)$$

$$R_m^{II}(z) = \frac{e^t N_m(z) e}{(1 - zb)(1 - zd)^m} .$$

Como estamos interessados em matrizes $B = D$ ou B nula , as representações acima mostram que ambas as funções de estabilidade são da forma :

$$R(z) = (1 - zd)^{-q} P(dz) , \quad P(dz) = \sum_{j=0}^r w_j (dz)^j , \quad (2.11)$$

onde os elementos w_j dependem de q e d .

Vamos listar na tabela abaixo os valores de q e r para os métodos dos Tipos A , B e C , de ambas as classes .

Tabela 2.1 - Valores de r e q que aparecem na função de estabilidade (2.11) .

Classe-Tipo	I A	I B	I C	II A	II B	II C
r	$m + 1$	$m + 1$	$m + 2$	m	m	$m + 1$
q	m	$m + 1$	$m + 1$	m	$m + 1$	$m + 1$

O nosso interesse é escolher m de modo que a ordem p do método PDIRK seja igual a r . A motivação para tal escolha é que , se o método predictor for $O(h)$, então , pelos teoremas 2.1 e 2.2 , a ordem p do método PDIRK não excede r .

Vamos desenvolver uma expressão para w_j (fórmula (2.11)) e assim obteremos uma representação explícita para as funções de estabilidade (2.10) .

Teorema 2.3 . Seja p a ordem de um método PDIRK arbitrário e seja m tal que $r = p$. Então os coeficientes w_j em (2.11) são dados por :

$$w_j = \sum_{i=0}^j \binom{q}{j-i} (-1)^{j-i} \frac{1}{i! d^i} , \quad j = 0, 1, \dots, q$$

$$w_j = \sum_{i=0}^q \binom{q}{i} (-1)^i \frac{1}{(j-i)! d^{j-i}} , \quad j = q+1, \dots, p .$$
(2.12)

Prova . Desde que assumimos que o método possui ordem p , nós temos necessariamente que $R(z) = \exp(z) + O(z^{p+1})$. Expandindo $(1 - dz)^q \exp(z)$ em série de Taylor , em torno de $z = 0$ e igualando os coeficientes da série com os coeficientes do polinômio $P(dz)$, definido em (2.11) , nós podemos achar os primeiros $p + 1$ coeficientes do polinômio P , que são exatamente os dados em (2.12) .

Observação : Como estamos supondo que $r = p$, devemos exigir que os métodos do Tipo C satisfaçam à condição $Ee = Ae - de$, pois dessa forma as ordens desses métodos se igualam a r (tabela 2.1) ; isto ocorre devido aos teoremas 2.1 e 2.2 .

Para estudarmos propriedades de estabilidade dos métodos PDIRK vamos definir o polinômio E .

$$E(u) = |(1 - iy)^q|^2 (1 - |R(-\frac{iy}{d})|^2) = |(1 - iy)^q|^2 - |P(iy)|^2 =$$

$$= (1 + u)^q - (w_0 - w_2 u + w_4 u^2 - \dots)^2 - u (w_1 - w_3 u + w_5 u^2 - \dots)^2,$$

onde $u = y^2$.

Sabendo que $R(z) = \exp(z) + O(z^{p+1})$ segue que

$$|R(-\frac{iy}{d})|^2 = 1 + O(y^{p+1}) ,$$

logo , $E(y^2) = O(y^{p+1})$. Portanto , todos os termos de $E(y^2)$ de grau menor que $p + 1$ em y são nulos . Nestas condições , para $q \geq p$, podemos escrever :

$$E(u) = \sum_{j=(p/2)+1}^q x_j u^j, \text{ onde } x_j = x_j(d) = \binom{q}{j} - w_j^2 - 2 \sum_{i=1}^j (-1)^i w_{j-i} w_{j+i},$$

$$w_j = 0 \text{ se } j > p \text{ ou } j < 0.$$

De fato,

quando j for par, o coeficiente de u^j é dado por

$$\binom{q}{j} - w_j^2 - 2 \left(\sum_{i=0}^{(j/2)-1} w_{2i} w_{2(j-i)-1} - \sum_{i=0}^{(1/2)(j-2)} w_{2i+1} w_{2(j-i)-1} \right),$$

e quando j for ímpar obtemos

$$\binom{q}{j} - w_j^2 - 2 \left(\sum_{i=0}^{(1/2)(j-1)-1} w_{2i+1} w_{2(j-i)-1} - \sum_{i=0}^{(1/2)(j-1)} w_{2i} w_{2(j-i)} \right).$$

Ambos os somatórios possuem j termos que podem ser ordenados da seguinte maneira:

j -par: as parcelas são dadas por

$$\alpha_{2n+1} = -w_{2i+1} w_{2(j-i)-1}, \text{ com } i = \frac{j}{2} - (n+1);$$

$$\alpha_{2n+2} = w_{2i} w_{2(j-i)}, \text{ com } i = \frac{j}{2} - (n+1).$$

j -ímpar: as parcelas são dadas por

$$\alpha_{2n+1} = -w_{2i} w_{2(j-i)}, \text{ com } i = \frac{j-1}{2} - n;$$

$$\alpha_{2(n+1)} = w_{2i+1} w_{2(j-i)-1}, \text{ com } i = \frac{j-1}{2} - (n+1).$$

Portanto, ambos os somatórios podem ser escritos como

$$\sum_{i=1}^j (-1)^i w_{j-i} w_{j+i}.$$

Notemos que A-estabilidade ocorre se e somente se $|R(iy)| \leq 1$, devido ao princípio do módulo máximo; então, um método PDIRK é A-estável se e somente se $E(u) \geq 0$, para todo $u \geq 0$.

Os valores de d para os quais " $R(z)$ é A-estável" serão chamados A-aceitáveis. Denotaremos por I_{pq} o conjunto dado por :

$$\{d : E(u) \geq 0, \text{ para todo } u \geq 0\}.$$

" $R(z)$ é L-estável" se for A-estável e se $q > p$. Métodos do Tipo B.1 e classe II possuem esta propriedade, como mostra a tabela 2.1. Podemos conseguir que um método A-estável se transforme num método L-estável, mas, para isto, devemos obter um valor d : A-aceitável, tal que w_p seja nulo, ou seja, o p -ésimo coeficiente do polinômio P , em (2.11), se anula.

Apresentaremos, na página seguinte, duas tabelas que contém valores de d : A-aceitáveis e L-aceitáveis. Estas tabelas foram extraídas de: [Houwen, Sommeijer, Couzy, 89], p. 7 e p. 8, que por sua vez as obteve de: [Wolfbrandt, 77].

Tabela 2.2 - Valores de d : A-aceitáveis e L-aceitáveis , para $p = q$.

$p = q$	I_{pp}	$d_{pp} (w_p = 0)$
1	$[\frac{1}{2}, \infty]$	1
2	$[\frac{1}{4}, \infty]$	$1 \pm \sqrt{(1/2)}$
3	$[\frac{1}{3}, 1.068]$	0.43586650
4	[0.395 , 1.280]	0.5728160625
5	[0.247 , 0.361] $\cup$ [0.421 , 0.473]	0.2780538410
6	[0.285 , 0.540]	0.3341423671
7	vazio	-----
8	[0.218 , 0.264]	0.2343731596
9	vazio	-----
10	vazio	-----

Tabela 2.3 - Valores de d : A-aceitáveis e L-aceitáveis ,
para $p = q - 1$.

$p = q-1$	$I_{p,p+1}$	$d_{p,p+1} (w_p = 0)$
1	$[1 - \sqrt{(1/2)}, 1 + \sqrt{(1/2)}]$	0.5
2	[0.181 , 2.185]	$0.5 \pm \sqrt{(1/12)}$
3	[0.224 , 0.572]	0.3025345782
4	[0.248 , 0.676]	0.3888576711
5	[0.184 , 0.334]	0.2168805435
6	[0.205 , 0.378]	0.2579552416
7	[0.157 , 0.2029] $\cup$ [0.2052 , 0.234]	0.1690246379
8	[0.171 , 0.259]	0.1929778040
9	vazio	-----
10	[0.147 , 0.165] $\cup$ [0.1938 , 0.1961]	0.1541460739 .

Quando $q = p - 1$, o que ocorre nos métodos dos Tipos A (exemplo 2) e C.2 (exemplo 4 , caso 2) , ambos da classe I , a função de estabilidade $R(z)$, em (2.11) , possui no numerador um polinômio de grau maior que o polinômio do denominador . Então uma condição necessária para A-estabilidade é anular o coeficiente w_p do polinômio P no numerador dessa função . Para $p = 2, \dots, 10$ foram investigados em [Wolfbrandt,77] , os possíveis valores de d que anulariam w_p , e a conclusão foi que métodos A-estáveis podem ser obtidos somente para : $p = 2$ e $d = 0.5$, $p = 3$ e $d = \frac{3 + \sqrt{3}}{6}$, $p = 4$ e $d = 1.0685790213$. Como estes métodos possuem ordem p (até 4) e $p - 1$ estágios sequenciais , a única vantagem de se implementar estes métodos em paralelo , ao invés de implementar um método DIRK sequencialmente , é que os PDIRK possuem métodos embutidos que controlam o comportamento do erro local .

Observação : É interessante notar que qualquer método s-RK (inclusive os explícitos) pode ser usado para gerar métodos PDIRK do Tipo B , classe I , A-estáveis , e se for "stiffly" preciso ele pode gerar métodos A-estáveis dos Tipos A e C da classe II , ou gerar métodos L-estáveis do Tipo B da classe II . Para este fim , basta encontrarmos os parâmetros A-aceitáveis e L-aceitáveis .

Exemplo 5 : Seja

$$y'(t) = f(y(t)) = \begin{bmatrix} -24 y_1 + 25 y_2 \\ 26 y_1 - 25 y_2 \end{bmatrix} , \quad y(0) = \begin{bmatrix} 1.0025 \\ 0.9974 \end{bmatrix} ,$$

$0 \leq t \leq 1$. A solução exata desse problema é :

$$y(t) = \frac{1}{51} (26 y_1(0) + 25 y_2(0)) e^t (1, 1)^t + \frac{1}{51} (y_1(0) - y_2(0)) e^{-50t} (25, -26)^t .$$

Este problema stiff foi proposto em [Butcher,Cash,89] , p. 6 . Vamos resolvê-lo numericamente por método PDIRK ; faremos um comentário a respeito do método Tipo B e classe II , ou seja , na fórmula (2.7) , a matriz E é nula e B = D e o método implementado tem que ser "stiffly" preciso .

Para resolvermos o PVI anterior , usaremos o método corretor Radau II A dois estágios , ordem três , "stiffly" preciso , que possui matriz

$$A = \begin{bmatrix} \frac{5}{12} & -\frac{1}{12} \\ \frac{3}{4} & \frac{1}{4} \end{bmatrix}$$

A matriz diagonal D foi escolhida com seus elementos iguais a $d = 0.3025345782$ (ver tabela 2.3) e dessa forma foi obtido um método PDIRK do Tipo B , classe II , L-estável .

Na implementação deste método é necessário o uso da matriz jacobiana da função f , dada por

$$(Jf)(y) = \begin{bmatrix} -24 & 25 \\ 26 & -25 \end{bmatrix} ,$$

que é usada nas iterações do método de Newton , o qual , neste exemplo , requer em cada passo , a solução de um sistema linear de ordem 4 (que é o número de equações implícitas do método 2-RK aplicado ao nosso problema de dimensão dois) . Porém , utilizando a técnica de iteração diagonal , nós mostramos que a matriz utilizada nas iterações de Newton é da forma bloco diagonal , com bloco diagonal igual a $I - h d J_f$ e isto possibilita transformar um sistema linear de ordem 4 em dois sistemas de ordem 2 . Como a implementação é feita em paralelo , estes dois sistemas têm o mesmo custo que um único sistema de ordem dois implementado sequencialmente .

No ponto $t = 1$, a solução exata do PVI é dada por $y_1(t) = y_2(t) = 2.718281828$. Utilizando o método PDIRK acima ,

obtivemos a seguinte tabela :

tamanho do passo	vetor solução	
h	$y_1(1)$	$y_2(1)$
0.5	2.7129814067	2.7129798365
0.05	2.7182776042	2.7182776042
0.005	2.7182818243	2.7182818243

A implementação deste método encontra-se no apêndice , programa "Método PDIRK" .

A tabela abaixo contém informações a respeito de métodos PDIRK com matriz diagonal de elementos iguais .

Tabela 2.4 - Propriedades de Métodos PDIRK com Matriz Diagonal de Elementos Iguais

Classe-Tipo	Ordem	Estágio Sequencial
I A	m+1	m
I B.1	m+1	m+1
I C.2	m+2	m+1
II A	m	m
II B.1	m	m+1
II C.2	m+1	m+1

2.3.3 - Métodos com Elementos Diagonais Arbitrários .

Vamos estudar agora a função de estabilidade (2.11) , considerando métodos PDIRK que possuem a matriz diagonal D (da iteração diagonal) com elementos diagonais arbitrários . Para isto , vamos reescrever a matriz Q_m que aparece em (2.9) :

$$\begin{aligned}
 Q_m &= (I - zD)^{-1}(I - Z^m) + Z^m Q_0 = \\
 &= (I - Z)^{-1}(I - Z^m) + Z^m(I - zD)(I - zB)^{-1}(I + zE) .
 \end{aligned}$$

Como $Z = z(A - D)(I - zD)^{-1}$, então $(I - zD)^{-1} = (I - zA)^{-1}(I - Z)$, logo

$$Q_m(z) = (I - zD)(I - zA)^{-1}[I - Z^m + (I - Z)Z^m(I - zD)(I - zB)^{-1}(I + zE)]$$

e podemos reescrever as funções de estabilidade dadas em (2.9) como :

$$R_m^I(z) = 1 + zb^t(I - zA)^{-1}[I - Z^m + (I - Z)Z^m(I - zD)(I - zB)^{-1}(I + zE)]e$$

$$R_m^{II}(z) = e_s^t(I - zA)^{-1}[I - Z^m + (I - Z)Z^m(I - zD)(I - zB)^{-1}(I + zE)]e = \\ = 1 + e_s^t(I - zA)^{-1}[zA - Z^m + (I - Z)Z^m(I - zD)(I - zB)^{-1}(I + zE)]e.$$

Quando nos referirmos a estas funções usaremos a numeração da fórmula abaixo :

$$\begin{array}{l} R_m^I(z) \\ R_m^{II}(z) \end{array} \quad (2.13)$$

O objetivo agora é obter representações para estas funções de estabilidade , que não envolvam inversas de matrizes , a menos que estas sejam matrizes diagonais .

Teorema 2.4 . As funções de estabilidade dadas em (2.13) podem ser representadas por

$$R_m^I(z) = \frac{\det\langle I - zA + z[I - Z^m + (I - Z)Z^m(I - zD)(I - zB)^{-1}(I + zE)]eb^t \rangle}{\det\langle I - zA \rangle}$$

$$R_m^{II}(z) = \frac{\det\langle I - zA + [zA - Z^m + (I - Z)Z^m(I - zD)(I - zB)^{-1}(I + zE)]ee_s^t \rangle}{\det\langle I - zA \rangle}$$

Prova . Basta usarmos a identidade (0.4.0) .

■

Corolário 2.4.1 . Seja a matriz dada por $Z = z(A - D)(I - zD)^{-1}$;
então valem :

a) As funções de estabilidade dos métodos PDIRK do Tipo A são dadas
por : (Tipo A $\Rightarrow E = B = 0$)

$$R_m^I(z) = \frac{\det(I - zA + z[I - zZ^m A]eb^t)}{\det(I - zA)} ,$$

$$R_m^{II}(z) = \frac{\det(I - zA + z[I - Z^m]Aee_s^t)}{\det(I - zA)} ,$$
(2.14)

b) As funções de estabilidade dos métodos PDIRK do Tipo B são dadas
por : (Tipo B $\Rightarrow E = 0$, $B = D$)

$$R_m^I(z) = \frac{\det(I - zA + z[I - Z^{m+1}]eb^t)}{\det(I - zA)} ,$$

$$R_m^{II}(z) = \frac{\det(I - zA + [zA - Z^{m+1}]ee_s^t)}{\det(I - zA)} ,$$
(2.15)

c) As funções de estabilidade dos métodos PDIRK do Tipo C.2 ou C.3 são
dadas por : (Tipo C.2 ou C.3 $\Rightarrow Ae = (E + B)e$, $B = D$)

$$R_m^I(z) = \frac{\det(I - zA + z[I - zZ^{m+1}A]eb^t)}{\det(I - zA)} ,$$

$$R_m^{II}(z) = \frac{\det(I - zA + z[I - Z^{m+1}]Aee_s^t)}{\det(I - zA)} .$$
(2.16)

■

Apresentamos a seguir uma tabela , a qual contém informações a
respeito de estabilidade de métodos PDIRK que possuem matriz diagonal D
arbitrária .

**Tabela 2.5 - Características de Estabilidade de Métodos PDIRK com
Matriz D (da iteração diagonal) Arbitrária .**

Classe-Tipo	Corretor	Ordem	Estagio seq.	Estabilidade
I B.2	Radau IIA	3	2	Fortemente A-estável
	Gauss-Legendre	4	3	Fortemente A-estável
	Radau IIA	5	4	Fortemente A-estável
	Gauss-Legendre	6	5	Fortemente A(α)-estável $\alpha = 89,97$ graus .
	Radau IIA	7	6	Fortemente A(α)-estável $\alpha = 89,3$ graus .
<hr/>				
II B.2	Radau IIA	3	3	L(α)-estável , $\alpha = 89,75$ graus .
	Radau IIA	5	5	L(α)-estável , $\alpha = 89,12$ graus .
	Radau IIA	7	7	L(α)-estável , $\alpha = 89,02$ graus .
<hr/>				
II C.3	Radau IIA	3	2	A-estável
	Radau IIA	5	4	A(α)-estável , $\alpha = 89,997$ graus
	Radau IIA	7	6	A(α)-estável , $\alpha = 89,95$.

Esta tabela foi extraída de [Houwen,Sommeijer,Couzy,89] , p. 11 .

Comparando os métodos da tabela acima com os métodos PDIRK obtidos na subseção 2.3.2 (tabela 2.4) , percebemos que os métodos do Tipo B.2 da classe I e os métodos C.3 da classe II precisam de um estágio sequencial a menos para obterem a mesma ordem de seus métodos correspondentes na subseção anterior . Mais ainda , com exceção do método Tipo B.2 de ordem sete , os métodos da tabela acima possuem boas propriedades de estabilidade como aqueles da subseção anterior . Os métodos do Tipo B.2 da classe II com ordem sete são os únicos que merecem atenção , pois na subseção anterior não foi possível construir um método L-estável de ordem sete com sete estágios sequenciais ; os métodos de ordens 3 e 5 do Tipo B.2 da classe II não são superiores

aos seus correspondentes na subseção anterior , porque naquela subseção foi possível construir métodos PDIRK , L-estáveis , com o mesmo número de estágios sequenciais desses métodos do Tipo B.2 da classe II e ordem 3 e 5 ; porém eles são de menor custo computacional .

2.4 - Métodos de Runge-Kutta Iterados até Convergência .

Os métodos que vamos estudar agora diferem dos métodos PDIRK que foram construídos a partir da limitação do número de iterações no processo (2.3) . Agora , o nosso objetivo é iterar (2.3) até convergência ; os parâmetros do método iterativo que pretendemos construir , devem ser escolhidos de modo que a solução do processo (2.3) convirja rapidamente para a solução do método de Runge-Kutta original (corretor) , e além disso proporcionem boas características de estabilidade , se possível tão boas quanto as do corretor . Nesta seção , vamos escolher parâmetros que aumentem a ordem de estágio (ver definição abaixo) dos métodos de Runge-Kutta ; isto evita o fenômeno de redução de ordem , observado em métodos DIRK , quando aplicados a problemas "stiff" : nestes casos a ordem de precisão de um método DIRK é reduzida a sua ordem de estágio (ou algumas vezes , ordem de estágio mais um) . Uma bibliografia adequada é [Hairer,Lubich,Roche,88] .

2.4.1 - Métodos de Runge-Kutta com Elevada Ordem de Estágio .

Com o objetivo de aumentar a ordem de estágio de métodos de Runge-Kutta , vamos passar a estudar métodos com a seguinte representação matricial :

$$\begin{array}{c|cc} 0 & 0 & 0^t \\ c & a & A \\ \hline & b_o & b^t \end{array} , \quad (2.17)$$

onde b_o é um escalar , a , b , c são vetores s -dimensionais e A é uma matriz $s \times s$.

Métodos desse tipo requerem aproximadamente o mesmo esforço computacional que métodos da forma :

$$\begin{array}{c|c} c & A \\ \hline & b^t \end{array} ,$$

porém (2.17) possui o vetor a de parâmetros que possibilita aumentar a ordem de estágio desses métodos . É fácil ver que , se $b_0 = 0$ e a for o vetor nulo , as equações dos dois métodos se igualam . Se os elementos acima não forem nulos , então as complexidades computacionais dos dois métodos diferem apenas em um estágio explícito a mais , que o método (2.17) tem que executar .

Quando o método (2.17) é aplicado a problemas escalares obtemos as seguintes fórmulas do método de Runge-Kutta :

$$\begin{aligned} Y &= y_{n-1} e + h a f(y_{n-1}) + h A F(Y) \\ y_n &= y_{n-1} + h b_0 f(y_{n-1}) + h b^t F(Y) \end{aligned} \quad (2.18)$$

Definição 2.2 : O método de Runge-Kutta (0.2) tem ordem de estágio r se

$$Y(t_n) - Y = O(h^{r+1}) ,$$

onde $Y(t_n)$ é o vetor que tem como componentes $y(t_{n-1} + h c_i)$ e y é localmente a solução exata do PVI (0.1) ; satisfazendo $y(t_{n-1}) = y_{n-1}$.

De forma equivalente , o método (2.18) tem ordem de estágio r se :

$$Y(t_n) - y_{n-1} e - h a f(y_{n-1}) - h A F(Y(t_n)) = O(h^{r+1}) .$$

As condições de ordem de estágio do método (2.18) , ou hipóteses simplificadoras de linha , como Butcher as chamou em [Butcher,87] , são dadas por :

$$C_1 = a + A e - c \quad ; \quad C_j = j A c^{j-1} - c^j \quad , \quad j = 2, 3, \dots$$

Então , ordem de estágio r é equivalente a $C_j = 0$, $j = 1, \dots, r$.

Observações :

1) As potências do vetor c são tomadas componente a componente .

2) Para obtermos ordem de estágio r , considerando o vetor de parâmetros c , nós necessitamos resolver sr equações lineares em $s(s+1)$ incógnitas , então , a máxima ordem de estágio obtida é $s+1$. O método de Runge-Kutta construído dessa forma será chamado de método de Lagrange .

O método (2.18) é "stiffly" preciso se satisfaz :

$$b_0 = e_s^t a \quad , \quad b^t = e_s^t A \quad . \quad (2.19)$$

Para analisar a estabilidade , aplicamos o método (2.18) ao problema teste escalar $y' = \lambda y$. Colocando $z = \lambda h$, obtemos :

$$Y = (I - zA)^{-1}(e + zA)y_{n-1} \quad ,$$

$$y_n = (1 + b_0 z)y_{n-1} + zb^t Y \quad .$$

Logo , escrevendo $y_n = R(z)y_{n-1}$, obtemos :

$$R(z) = 1 + b_0 z + zb^t(I - zA)^{-1}(e + zA) \quad . \quad (2.20)$$

No caso de métodos "stiffly" precisos , ou seja , métodos que satisfazem (2.19) , a função de estabilidade $R(z)$ é dada por :

$$R(z) = e_s^t (I - zA)^{-1}(e + za) \quad (2.21)$$

Estas funções de estabilidade serão usadas na próxima subseção , quando formos estudar estabilidade de métodos iterativos diagonais .

2.4.2 - Iteração Diagonal Aplicada a Métodos RK com Elevada Ordem de Estágio .

Nós vamos usar iteração diagonal para resolver as equações implícitas do método (2.18) . A iteração é dada por :

$$Y^{(1)} - hD F(Y^{(1)}) = y_{n-1}e + ha f(y_{n-1}) + h(A - D)F(Y^{(0)}) \quad , \quad (2.22a)$$

$$Y^{(k)} - hD F(Y^{(k)}) = y_{n-1}e + ha f(y_{n-1}) + h(A - D)F(Y^{(k-1)}) \quad ,$$

$k = 2, 3, \dots$.

A aproximação inicial $Y^{(0)}$, para o vetor Y , será sempre dada por :

$$Y^{(0)} = y_{n-1} e .$$

Se m iterações são executadas, definimos

1) $y_n = y_{n-1} + hb_0 f(y_{n-1}) + hb^t F(Y^{(m)})$, para métodos não "stiffly" precisos ,

(2.22b)

2) $y_n = e_e^t Y^{(m)}$, para métodos "stiffly" precisos .

Com esta definição, o método formado pelas equações (2.22a,b) pertence à classe dos métodos DIRK. A sua representação matricial é dada abaixo :

$k = 0$	0					
$k = 1$	$c - De$	D				
$k = 2$	a	$A - D$	D			
$k = 3$	a	0	$A - D$	D		
.....						
$k = m$	a	0	...	0	$A - D$	D
1)	b_0	0^t	...	0^t	0^t	b^t
2)	$e_e^t a$	0^t	...	0^t	$e_e^t (A - D)$	$e_e^t D$

onde 1) é a representação para métodos não stiffly precisos, e 2) é a representação para métodos stiffly precisos.

Observação : Estamos supondo que o método (2.18) tem ordem de estágio no mínimo igual a 1 : isto significa que a seguinte relação é válida : $Ae + a = c$, então o vetor $c - De$ que aparece na representação acima, é consequência dessa suposição e do fato de que no caso escalar vale a igualdade : $F(Y^{(0)}) = f(y_{n-1})e$.

- Ordem de Precisão do Método Iterativo Diagonal .

Vamos analisar a diferença entre $Y(t_n)$ e $Y^{(k)}$, onde $Y(t_n)$ é o vetor dado na definição 2.2, e $Y^{(k)}$ é dado no processo iterativo (2.22a).

Temos

$$\begin{aligned} Y(t_n) - Y^{(k)} &= (Y(t_n) - Y) + (Y - Y^{(k)}) = \\ &= (Y(t_n) - Y) + hD [F(Y) - F(Y^{(k)})] + h(A - D)[F(Y) - F(Y^{(k-1)})] \end{aligned}$$

e usando o teorema do valor médio obtemos

$$Y - Y^{(k)} = hD J_{1F} (Y - Y^{(k)}) + h(A - D) J_{2F} (Y - Y^{(k-1)}),$$

onde $J_{1F} = J_F(Y^{(k)} + \theta_1(Y - Y^{(k)}))$, $J_{2F} = J_F(Y^{(k-1)} + \theta_2(Y - Y^{(k-1)}))$, com $\theta_1, \theta_2 \in (0,1)$ (lembramos que J_F é a matriz jacobiana da função F do método de Runge-Kutta).

Considerando uma aproximação de $O(h)$, para os dois jacobianos acima podemos escrever :

$$Y - Y^{(k)} = h(I - hD J_F)^{-1} (A - D) J_F (Y - Y^{(k-1)}).$$

Agora, definindo a matriz de recursão :

$$Z = h(I - hD J_F)^{-1} (A - D) J_F,$$

obtemos

$$Y - Y^{(k)} = Z^k (Y - Y^{(0)}),$$

e assim

$$Y(t_n) - Y^{(k)} = (Y(t_n) - Y) + Z^k (Y - Y^{(0)}), \quad k = 1, 2, \dots$$

Se o método (2.18) tiver ordem de estágio r , ou seja, $Y(t_n) - Y = O(h^{r+1})$, então temos

$$Y(t_n) - Y^{(k)} = O(h^{r+1}) + O(h^{k+1}), \quad (2.23)$$

desde que $Z = O(h)$ e $Y^{(0)} = y_{n-1}$.

Então, depois de m iterações, (2.22a,b) define um método no qual $Y^{(m)}$ aproxima $Y(t_n)$ com ordem de estágio $r^* = \min(r, m)$.

- Estabilidade do Método Iterativo Diagonal.

Com o objetivo de vermos como a função de estabilidade do método (2.22a,b) se comporta depois de m iterações, nós aplicamos este método ao problema teste escalar $y' = \lambda y$. Colocando $z = h\lambda$, deduzimos :

$$\begin{aligned}
Y^{(m)} &= zDY^{(m)} + y_{n-1}e + zay_{n-1} + z(A - D)Y^{(m-1)} \\
Y^{(m)} &= (I - zD)^{-1}[(e + za)y_{n-1} + z(A - D)Y^{(m-1)}] = \\
&= ZY^{(m-1)} + (I - zD)^{-1}(e + za)y_{n-1} ,
\end{aligned}$$

onde Z assume a forma $Z = z(I - zD)^{-1}(A - D)$. Podemos ainda escrever

$$\begin{aligned}
Y^{(m)} &= Z^m Y^{(0)} + (I + Z + \dots + Z^{m-1})(I - zD)^{-1}(e + za)y_{n-1} \\
Y^{(m)} &= [Z^m e + (I - Z)^{-1}(I - Z^m)(I - zD)^{-1}(e + za)]y_{n-1} . \quad (2.24)
\end{aligned}$$

Discutiremos, separadamente, os casos de iterações dos métodos (2.18) que não são "stiffly" precisos e os que são "stiffly" precisos:

1) Métodos não stiffly precisos:

Seja $y_n = (1 + zb_0)y_{n-1} + zb^t Y^{(m)}$; usando a expressão de $Y^{(m)}$ obtida acima, podemos reescrever y_n como

$$y_n = (1 + zb_0 + zb^t[Z^m e + (I - Z)^{-1}(I - Z^m)(I - zD)^{-1}(e + za)])y_{n-1} ,$$

e a função de estabilidade é dada por

$$R_m(z) = 1 + zb_0 + zb^t[Z^m e + (I - Z)^{-1}(I - Z^m)(I - zD)^{-1}(e + za)] .$$

Como $Z = (I - zD)^{-1}(zA - zD)$, então $(I - zD)^{-1} = (I - Z)(I - zA)^{-1}$; assim podemos reescrever

$$R_m(z) = 1 + zb_0 + zb^t(I - zA)^{-1}(e + za) - z^z b^t Z^m (I - zA)^{-1}(Ae + a) .$$

Como nós assumimos que a ordem de estágio de (2.18) é pelo menos 1, vale a igualdade $Ae + a = c$, logo

$$R_m(z) = R(z) - z^z b^t Z^m (I - zA)^{-1}c , \text{ onde } R(z) \text{ é a função (2.20) .}$$

Substituindo a expressão de Z na fórmula acima obtemos:

$$R_m(z) = R(z) - z^{m+z} b^t [(I - zD)^{-1}(A - D)]^m (I - zA)^{-1} .$$

Através dessa expressão nós podemos analisar o comportamento de $R_m(z)$ para valores grandes de $|z|$. Então, quando $|z| \rightarrow \infty$ temos

$$R_m(z) = R(z) + zb^t(I - D^{-1}A)^m A^{-1}c . \quad (2.25)$$

e isto mostra que, para qualquer valor fixo de m , a função de estabilidade é ilimitada quando $|z| \rightarrow \infty$. Portanto nós concluímos que

métodos de Runge-Kutta , não "stiffly" precisos , não devem ser iterados até convergência , usando iteração diagonal .

2) Métodos "stiffly" precisos :

Seja $y_n = e_0^t Y^{(m)}$. Usando (2.24) , a função de estabilidade correspondente a este método é :

$$R_m(z) = e_0^t [Z^m e + (I - Z)^{-1}(I - Z^m)(I - zD)^{-1}(e + za)] .$$

É fácil ver que esta função pode ser reescrita como

$$\begin{aligned} R_m(z) &= e_0^t [Z^m e + (I - zA)^{-1}(e + za) - Z^m(I - zA)^{-1}(e + za)] = \\ &= R(z) - e_0^t [Z^m(I - zA)^{-1}(za + zAe)] \end{aligned}$$

$$R_m(z) = R(z) - z^{m+1} e_0^t [(I - zD)^{-1}(A - D)]^m (I - zA)^{-1} e ,$$

onde $R(z)$ é definida por (2.21) .

Agora , escrevemos :

$$R_m(z) = R(z) - [\sigma_m(z)]^m ,$$

$$\text{onde } \sigma_m(z) = [z^{m+1} e_0^t [(I - zD)^{-1}(A - D)]^m (I - zA)^{-1} e]^{(1/m)} .$$

Nós chamaremos $\sigma_m(z)$ o *fator de convergência* associado a z . Por exemplo , nós temos que $\sigma_m(0) = 0$ e $\sigma_m(\infty) = [-e_0^t (I - D^{-1}A)^m A^{-1} e]^{(1/m)}$.

O nosso objetivo é minimizar o fator de convergência no infinito , ou seja , minimizar $\sigma_m(\infty)$. Baseado nesta idéia construiremos métodos iterativos diagonais "stiffly" precisos com boas propriedades de estabilidade .

- Construção de Métodos Iterativos Diagonais .

Vamos considerar D uma matriz diagonal com elementos positivos , então dado um método s-RK nós vamos minimizar o raio espectral $\rho(I - D^{-1}A)$, sobre todas as possíveis matrizes D com elementos positivos .

Nós construiremos métodos da forma (2.17) , com $s = 2$, porque tais métodos podem ser derivados através de cálculos analíticos ; para $s > 2$ são necessários recursos numéricos . O método iterativo é obtido conseguindo-se uma matriz diagonal conveniente , a qual vai ser achada efetuando-se $\rho(I - D^{-1}A) = 0$, onde A é a matriz obtida do método (2.17) .

- Métodos de Lagrange .

Os métodos de Lagrange com matriz A de ordem $s = 2$ são determinados resolvendo-se o seguinte sistema linear :

$$Ae + a = c$$

$$jAc^{j-1} = c^j, \quad j = 2, 3, \dots,$$

onde A é uma matriz 2×2 , a é um vetor bi-dimensional, $c = (c_1, 1)^t$ e as potências do vetor c são consideradas componente a componente. Este sistema representa as condições que o método precisa satisfazer para que atinja ordem de estágio igual a três. A solução do sistema acima fornece :

$$A = \frac{1}{6(1 - c_1)} \begin{bmatrix} c_1(3 - 2c_1) & -c_1^2 \\ c_1^{-1} & 2 - 3c_1 \end{bmatrix} ;$$

$$a = \frac{1}{6(1 - c_1)} \begin{bmatrix} 3c_1 - 4c_1^2 + c_1^3 \\ -c_1^{-1} + 4 - 3c_1 \end{bmatrix}$$

Usando a fórmula (2.21) obtemos a seguinte função de estabilidade :

$$R(z) = \frac{6 + 2(2 - c_1)z + (1 - c_1)z^2}{6 - 2(c_1 + 1)z + cz^2}.$$

Então $|R(z)| \leq 1$, se e somente se $c_1 \geq \frac{1}{2}$.

Para determinarmos a matriz D é conveniente escrever :

$$D = \frac{1}{6(1 - c_1)} \begin{bmatrix} \frac{1}{d_1} & 0 \\ 0 & \frac{1}{d_2} \end{bmatrix}, \quad \text{logo}$$

$$I - D^{-1}A = \begin{bmatrix} 1 - c_1(3 - 2c_1)d_1 & c_1^2 d_1 \\ -c_1^{-1} d_1 & 1 - (2 - 3c_1)d_2 \end{bmatrix}.$$

Os autovalores de $I - D^{-1}A$ satisfazem a equação :

$$u^2 - Su + P = 0,$$

$$S = \text{Tr}(A) = 2 - c_1(3 - 2c_1)d_1 - (2 - 3c_1)d_2,$$

$$P = \det A = [1 - c_1(3 - 2c_1)d_1][1 - (2 - 3c_1)d_2] + c_1^2 d_1 d_2.$$

Colocando $S = 0 = P$ nós obtemos $\rho(I - D^{-1}A) = 0$. Nestas condições os parâmetros d_1, d_2 satisfazem :

$$c_1(3 - 2c_1)d_1 + (2 - 3c_1)d_2 = 2 \quad \text{e} \quad [1 - c_1(3 - 2c_1)d_1]^2 - c_1^2 d_1 d_2 = 0,$$

portanto

$$d_1 = \frac{1 + Q}{c_1(3 - 2c_1)}, \quad d_2 = \frac{1 - Q}{2 - 3c_1}, \quad Q = \pm \frac{\sqrt{6c_1}}{6(1 - c_1)}.$$

Observações :

1) Nós vimos que a função de estabilidade $R_m(z)$, do método iterativo diagonal, converge para a função de estabilidade do método de Runge-Kutta "stiffly" preciso, então é interessante que este método possua boas propriedades de estabilidade. No caso de métodos de Lagrange, nós temos a seguinte conjectura :

" Uma condição necessária para um método de Lagrange, "stiffly" preciso, ser fortemente A-estável é : $\sum_{j=1}^s c_j > \frac{s+1}{2}$, onde s é o número de estágios ". [Houwen, Sommeijer, 90]

2) Não podemos obter um método de Lagrange L-estável, no caso $s = 2$, porque isto só ocorreria se $c_1 = 1$, implicando na impossibilidade de construção da matriz A desse método.

Métodos de Newton-Cotes

Os métodos de Newton-Cotes são obtidos da mesma forma que os métodos de Lagrange, só que o vetor c tem componentes $c_j = \frac{j}{s}$.

Então o método de Newton-Cotes derivado do método de Lagrange é obtido colocando-se $c_1 = \frac{1}{2}$. Este valor de c_1 ocasiona um método A-estável.

Exemplo 6 : Vamos exibir um método de Lagrange que possui ordem de estágio $r = 3$, que também é a sua ordem global. Escolhendo $c_1 = -\frac{3}{4}$, nós obtemos :

$$A = \frac{1}{288} \begin{bmatrix} 216 & -81 \\ 256 & -48 \end{bmatrix}, \quad a = \frac{1}{288} \begin{bmatrix} 81 \\ 80 \end{bmatrix}, \quad c = \begin{bmatrix} \frac{3}{4} \\ 1 \end{bmatrix}.$$

A versão iterativa desse método é obtida acrescentando-se a matriz

$$D = \begin{bmatrix} \frac{3}{4(\sqrt{2} + 1)} & 0 \\ 0 & \frac{1}{6(\sqrt{2} - 1)} \end{bmatrix}.$$

Exemplo 7 : Usando o método de Lagrange, proposto no exemplo anterior, nós resolvemos o PVI do exemplo 5 (capítulo 2), através da técnica de iteração diagonal até convergência. A tabela abaixo contém os resultados obtidos em $t = 1$.

tamanho do passo	vetor solução	
h	$y_1(t)$	$y_2(t)$
0.5	2.7156113201	2.7154507972
0.05	2.7182794349	2.7182794349
0.005	2.7182818261	2.7182818261

A implementação deste método encontra-se no apêndice, programa "Método PDIRK Até Convergência".

Lembramos que a solução exata é $y_1(1) = y_2(1) = 2.718281828$.

2.5 - Conclusão .

Vimos neste capítulo a técnica de iteração diagonal sob o ponto de vista da implementação paralela . Esta técnica facilita a implementação de métodos de RK completamente implícitos . Isto ocorre porque os métodos iterativos resultantes têm o mesmo efeito de um método diagonalmente implícito , ou seja , em cada estágio uma única equação implícita , em uma única incógnita , é resolvida num processador .

Observamos que os métodos obtidos dessa técnica são superiores aos métodos DIRK , já que estes possuem ordem limitada por $s+1$, enquanto que os métodos PDIRK possuem a mesma ordem dos seus corretores .

Os experimentos numéricos apresentados nos exemplos 5 e 7 comprovam a eficiência de alguns métodos estudados .

MÉTODO DO TIPO I

```
#USE userio
#USE userval
#USE interf
#USE dblmath

VAL eps IS 0.0000000001(REAL64) : --TOLERANCIA PARA O ERRO

VAL t0 IS 0.0(REAL64) :      --EXTREMO INFERIOR DO INTERVALO [t0,T] QUE
                             -- SERA DISCRETIZADO EM PONTOS IGUALMENTE
                             -- ESPACADOS (METODO DE RK TIPO I)

VAL T IS 1.0 (REAL64) :      -- EXTREMO SUPERIOR DO INTERVALO [t0,T]
                             -- VAMOS CALCULAR y(T) APLICANDO METODO DE R
                             -- DE 4 ESTAGIOS (TIPO I) AO PVI dy/dt = f(t
                             -- y(t0) = y0

VAL y0 IS 0.1(REAL64) :      -- VALOR INICIAL DO PVI

{{{ PROC P1
PROC P1 (CHAN OF REAL64 in , out , ch , VAL REAL64 t0 , T , y0 ,
        eps , REAL64 h)
  VAL REAL64 um IS 1.0(REAL64) :
  VAL REAL64 dois IS 2.0(REAL64) :
  VAL REAL64 TR IS 3.0(REAL64) :
  VAL REAL64 cin IS 5.0(REAL64) :
  VAL REAL64 seis IS 6.0(REAL64) :
  VAL REAL64 doz IS 12.0 (REAL64) :
  REAL64 Y1 , Y2 , y , indj , mod.x , den , ex , ey , comp.int :

  [3] REAL64 x :      --VETOR QUE CONTROLA AS ITERACOES DO METODO DE NEWT

  [3][3] REAL64 a : --MATRIZ DO METODO DE RK DO TIPO I (DEFINIDA EM P1)

  REAL64 A , B , Ax , Ay , Bx , By , c1 , c2 , dif , tn , aux :

  [5]REAL64 c : --VETOR DE NOS DE UM METODO DE RK DE 4 ESTAGIOS
  [5]REAL64 t : --ESTE VETOR ARMAZENA OS VALORES DE t + hc[i]
                -- n-1

  --DECLARACOES DE FUNCOES

  REAL64 FUNCTION DSQRT (VAL REAL64 X)      -- FUNCAO RAIZ QUADRADA
    REAL64 RAIZ.QUADRADA :
    VALOF
      RAIZ.QUADRADA := DSQRT(X)
    RESULT RAIZ.QUADRADA
  :

  REAL64 FUNCTION DEXP (VAL REAL64 X)      -- FUNCAO EXPONENCIAL
    REAL64 EXPON :
    VALOF
      EXPON := DEXP(X)
    RESULT EXPON
  :
```

```

REAL64 FUNCTION f ( VAL REAL64 t , y )      -- FUNCAO DO PVI  $dy/dt = f(t,y)$ 
                                           --  $y(t_0) = y_0$ 
  VAL REAL64 DEZ IS 10.0 (REAL64) :
  REAL64 FUN , ARG :
  VALOF
    SEQ
      ARG := DEZ*(t - y)
      FUN := DEXP(ARG)
  RESULT FUN
:

REAL64 FUNCTION df (VAL REAL64 t , y)      -- DERIVADA DA FUNCAO f
  REAL64 Df , ARG :
  VAL DEZ IS 10.0(REAL64) :
  VALOF
    SEQ
      ARG := DEZ*(t - y)
      Df := - (DEZ*(DEXP(ARG)))
  RESULT Df
:

SEQ
  PAR
    a[1][1] := cin/doz
    a[2][1] := (um/doz) + ((DSQRT(TR))/seis)
    a[1][2] := (um/doz) - ((DSQRT(TR))/seis)
    a[2][2] := cin/doz
    c[1] := (um/dois) - ((DSQRT(TR))/seis)
    c[2] := (um/dois) + (DSQRT(TR)/seis)
  PAR
    out ! h  --ENVIO DO TAMANHO DO PASSO PARA O PROCEDIMENTO P2
    ch ! h   --ENVIO DO TAMANHO DO PASSO PARA O PROCEDIMENTO WRITE
  y := y0    -- y0 = y(t0) VALOR INICIAL DO PVI
  tn := t0   --EXTREMO INFERIOR DO INTERVALO [t0,T]
  comp.int := T - t0  -- COMPRIMENTO DO INTERVALO [t0,T]
  indj := 1,0(REAL64)
  WHILE (indj*h) <= (comp.int + eps)
    SEQ
      PAR i = 1 FOR 2
        t[i] := tn + (h*c[i])
      x[1] := y
      x[2] := y
      mod.x := 10.0 (REAL64)

```

```
--METODO DE NEWTON PARA OBTEN OS VALORES DE Y1 E Y2
--DO METODO DE RK DO TIPO I
```

```
WHILE mod.x > eps
  SEQ
```

```
    PAR
      c1 := (a[1][1]*f(t[1] , x[1])) + (a[1][2]*f(t[2] , x[2]))
      c2 := (a[2][1]*f(t[1] , x[1])) + (a[2][2]*f(t[2] , x[2]))
    PAR
      A := (x[1] - (h*c1)) - y
      B := (x[2] - (h*c2)) - y
      Ax := um - ((a[1][1]*df(t[1] , x[1]))*h)
      Ay := - ((a[1][2]*df(t[2] , x[2]))*h)
      Bx := - ((a[2][1]*df(t[1] , x[1]))*h)
      By := um - ((a[2][2]*df(t[2] , x[2]))*h)
      den := (Ax*By) - (Ay*Bx)
    PAR
      ex := ((B*Ay) - (A*By))/den
      ey := ((A*Bx) - (B*Ax))/den
    PAR
      x[1] := x[1] + ex
      x[2] := x[2] + ey
      mod.x := (ex*ex) + (ey*ey)
      mod.x := DSQRT(mod.x)
```

```
--VALORES DE Y1 E Y2 OBTIDOS DO METODO DE NEWTON
```

```
PAR
  Y1 := x[1]
  Y2 := x[2]
```

```
--CALCULO DA APROXIMACAO PARCIAL DE y(T) (PROCEDIMENTO P1)
```

```
y := y + (h*((um/dois)*(f(t[1] , Y1) + f(t[2] , Y2))))
```

```
in ? aux          --RECEBENDO DO PROCEDIMENTO P2 A APROXIMACAO
                  --PARCIAL DE y(T)
```

```
y := (TR*y) - (dois*aux)    --CALCULO DA APROXIMACAO DE y(T)
```

```
out ! y          --ENVIO DA APROXIMACAO DE y(T) PARA
                  --O PROCEDIMENTO P2
```

```
tn := tn + h
indj := indj + um
```

```
ch ! y          --ENVIO DA SOLUCAO DO PVI PARA
                  --O PROCEDIMENTO WRITE
```

```
}}}
```

```

{{{ PROC P2
PROC P2 ( CHAN OF REAL64 in , out , VAL REAL64 t0 , T , y0 , eps)
  VAL REAL64 um IS 1.0(REAL64) :
  VAL REAL64 dois IS 2.0(REAL64) :
  VAL REAL64 tr IS 3.0(REAL64) :
  VAL cin IS 5.0(REAL64) :
  VAL REAL64 seis IS 6.0(REAL64) :

  REAL64 h :    --ESTA VARIÁVEL CONTROLA O TAMANHO DO PASSO DO METODO DE RK

  [3] REAL64 c :  -- VETOR DE NOS DO METODO DE RK DO TIPO I

  [3] REAL64 t :  --VETOR QUE ARMAZENA OS VALORES DE t      + hc[i]
                                     --n-1

  REAL64 Y1 , Y2 , y , ex , ey , mod.x , aux , indj :

  [3]REAL64 x :    --CONTROLA AS ITERACOES DO METODO DE NEWTON

  [3][3]REAL64 a :  -- MATRIZ DO METODO DE RK DO TIPO I (DEFINIDA EM P2)

  REAL64 A , B , Ax , Ay , Bx , By , c1 , c2 , DEN , tn , comp.int :

  --DEFINICOES DE FUNCOES

  REAL64 FUNCTION DSQRT ( VAL REAL64 X)    --FUNCAO RAIZ QUADRADA
    REAL64 RAIZ.QUADRADA :
    VALOF
      RAIZ.QUADRADA := DSQRT(X)
    RESULT RAIZ.QUADRADA
  :

  REAL64 FUNCTION DEXP (VAL REAL64 X)      --FUNCAO EXPONENCIAL
    REAL64 EXP :
    VALOF
      EXP := DEXP(X)
    RESULT EXP
  :

  REAL64 FUNCTION f ( VAL REAL64 t , y )  --FUNCAO DO PVI dy/dt = f(t,y)
                                           -- y(t0) = y0

    VAL REAL64 DEZ IS 10.0(REAL64) :
    REAL64 FUN , ARG :
    VALOF
      SEQ
        ARG := DEZ*(t - y)
        FUN := DEXP(ARG)
      RESULT FUN
  :

```

```

REAL64 FUNCTION df ( VAL REAL64 t , y)      --DERIVADA DA FUNCAO f
REAL64 Df , ARG :
VAL DEZ IS 10.0(REAL64) :
VALOF
SEQ
    ARG := DEZ*(t - y)
    Df := - (DEZ*DEXP(ARG))
RESULT Df
:

SEQ
PAR
    a[1][1] := 0.5(REAL64)
    a[1][2] := - ((DSQRT(tr))/seis)
    a[2][1] := (DSQRT(tr))/seis
    a[2][2] := 0.5 (REAL64)
    c[1] := (um/DOIS) - (DSQRT(tr)/seis)
    c[2] := (um/DOIS) + (DSQRT(tr)/seis)

out ? h      --RECEBIMENTO DO TAMANHO DO PASSO DO PROCEDIMENTO P1

y := y0      -- y0 = y(t0) VALOR INICIAL DO PVI

tn := t0     --EXTREMO INFERIOR DO INTERVALO [t0,T]

comp.int := T - t0      -- COMPRIMENTO DO INTERVALO [t0,T]

indj := um

WHILE (indj*h) <= (comp.int + eps)
SEQ
PAR
    x[1] := y + (um/cin)
    x[2] := y + (um/cin)
PAR i = 1 FOR 2
    t[i] := tn + (h*c[i])
mod.x := 10.0(REAL64)

--METODO DE NEWTON PARA OBTERMOS Y1 E Y2 DO METODO DE RK DO TIPO I

WHILE mod.x > eps
SEQ
PAR
    c1 := (a[1][1]*f(t[1] , x[1])) + (a[1][2]*f(t[2] , x[2]))
    c2 := (a[2][2]*f(t[2] , x[2])) + (a[2][1]*f(t[1] , x[1]))
PAR
    A := (x[1] - (h*c1)) - y
    B := (x[2] - (h*c2)) - y
    Ax := um - ((a[1][1]*df(t[1] , x[1]))*h)
    Ay := -((a[1][2]*df(t[2] , x[2]))*h)
    Bx := - ((a[2][1]*df(t[1] , x[1]))*h)
    By := um - ((a[2][2]*df(t[2] , x[2]))*h)
DEN := (Ax*By) - (Ay*Bx)

```

```

    PAR
        ex := ((B*Ay) - (A*By))/DEN
        ey := ((A*Bx) - (B*Ax))/DEN
    PAR
        x[1] := x[1] + ex
        x[2] := x[2] + ey
        mod.x := (ex*ex) + (ey*ey)
        mod.x := DSQRT(mod.x)

    --VALORES DE Y1 E Y2 OBTIDOS PELO METODO DE NEWTON

    PAR
        Y1 := x[1]
        Y2 := x[2]

    --CALCULO DA APROXIMACAO PARCIAL DE y(T) (PROCEDIMENTO P2)

    y := y + ((h/DOIS)*(f(t[1] , Y1) + f(t[2] , Y2)))

    in ! y -- ENVIO DA APROXIMACAO PARCIAL DE y(T) PARA
        -- O PROCEDIMENTO P1

    out ? y --RECEBENDO DO PROCEDIMENTO P1 A APROXIMACAO
        --DE y(T) .

    tn := tn + h
    indj := indj + um
:
}}
{{{ PROC WRITE
PROC WRITE (CHAN OF INT keyboard , CHAN OF ANY screen , CHAN OF REAL64 ch)
    REAL64 y , h :
    INT cont :
    SEQ
        newline (screen)
        newline (screen)
        write.full.string (screen , "PARA h = ")

    ch ? h --RECEBENDO O TAMANHO DO PASSO DO PROCEDIMENTO P1

    write.real64 (screen , h , 2 , 10)
    newline (screen)
    newline (screen)
    write.full.string (screen , "A SOLUCAO DO PVI E DADA POR ")
    newline (screen)
    newline (screen)

    ch ? y --RECEBENDO DO PROCEDIMENTO P1 A SOLUCAO DO PVI

    write.real64 (screen , y , 2 , 10)
    newline (screen)
    newline (screen)
:
}}

```


MÉTODO DO TIPO II

```
#USE userio
#USE uservals
#USE interf
#USE dblmath
```

```
VAL eps IS 0.0000000001 (REAL64) : --TOLERANCIA DO ERRO

VAL y0 IS 0.1 (REAL64) : -- VALOR INICIAL DO PVI

VAL t0 IS 0.0 (REAL64) : --EXTREMO INFERIOR DO INTERVALO [t0,T]
--QUE SERA DESCRETIZADO POR PONTOS
--IGUALMENTE ESPACADOS .(METODO DE RK)

VAL REAL64 T IS 1.0(REAL64) : --EXTREMO SUPERIOR DO INTERVALO [t0,T]
--VAMOS DBTER UMA APROXIMACAO PARA y(T)
--UTILIZANDO METODO DE RK DO TIPO II
--E 4 ESTAGIOS .
```

```
{{{ PROC P1
```

```
PROC P1 (CHAN OF REAL64 in , out , ch , VAL REAL64 t0 , T , y0 ,
      eps , REAL64 h )
```

```
VAL UM IS 1.0(REAL64) :
VAL DOIS IS 2.0 (REAL64) :
VAL TR IS 3.0 (REAL64) :
VAL CIN IS 5.0 (REAL64) :
VAL QT IS 4.0 (REAL64) :
VAL MIL IS 1000.0 (REAL64) :
VAL DEZ IS 10.0 (REAL64) :
REAL64 Y1 , Y2 , Y3 , Y4 :
REAL64 x , xaux , mo.dif , const1 , indj , y , tn :
REAL64 dif , A , Ax , f1 , f2 , f3 , f4 , comp.int :
```

```
[5] REAL64 c : -- VETOR DE NOS DO METODO DE RK DE 4 ESTAGIOS (TIPO II)
```

```
[5] REAL64 t : -- VETOR QUE ARMAZENA t + h*c[i] ( n-esimo passo)
--n-1
```

```
REAL64 FUNCTION DSQRT ( VAL REAL64 X ) --FUNCAO RAIZ QUADRADA
REAL64 RAIZ.QUADRADA :
VALOF
RAIZ.QUADRADA := DSQRT(X)
RESULT RAIZ.QUADRADA
:
```

```
REAL64 FUNCTION DALOG (VAL REAL64 X) --FUNCAO LOGARITMO NEFERIANO
REAL64 LN :
VALOF
LN := DALOG(X)
RESULT LN
:
```

```
REAL64 FUNCTION DEXP (VAL REAL64 X) --FUNCAO EXPONENCIAL
REAL64 EXPON :
VALOF
EXPON := DEXP(X)
RESULT EXPON
:
```

```

REAL64 FUNCTION f (VAL REAL64 t , y )      -- FUNCAO DO PVI  $dy/dt = f(t,y)$ 
                                           --  $y(t_0) = y_0$ 
REAL64 FUN , ARG :
VAL REAL64 DEZ IS 10.0(REAL64) :
VALOF
  SEQ
    ARG := DEZ*(t - y)
    FUN := DEXP(ARG)
  RESULT FUN
:

REAL64 FUNCTION df ( VAL REAL64 t , y )  --DERIVADA DA FUNCAO f
REAL64 Df , ARG :
VAL REAL64 DEZ IS 10.0(REAL64) :
VALOF
  SEQ
    ARG := DEZ*(t - y)
    Df := -(DEZ*(DEXP(ARG)))
  RESULT Df
:

SEQ
  PAR
    c[1] := UM/DOIS
    c[2] := DOIS/TR
    c[3] := UM/DOIS
    c[4] := UM/TR
  PAR
    out ! h      --ENVIIO DO TAMANHO DO PASSO PARA O PROCEDIMENTO P2
    ch ! h       --ENVIIO DO TAMANHO DO PASSO PARA O PROCEDIMENTO WRITE
    y := y0      -- VALOR INICIAL DO PVI
    tn := t0     -- EXTREMO INFERIOR DO INTERVALO [t0,T]
    comp.int := T - t0      -- COMPRIMENTO DO INTERVALO [t0,T]
    indj := 1.0(REAL64)
    WHILE (indj*h) <= (comp.int + eps)
      SEQ
        PAR i = 1 FOR 4
          t[i] := tn + (h*c[i])
          --CALCULO DE Y1 (Metodo de Newton)
          x := y
          mo.dif := 10.0(REAL64)
          WHILE mo.dif > eps
            SEQ
              xaux := x
              PAR
                A := x - (((h/DOIS)*f(t[i] , x)) + y)
                Ax := UM - ((h/DOIS)*df(t[i] , x))
              x := x - (A/Ax)
              dif := x - xaux
              mo.dif := DSQRT((dif*dif))

```

```

Y1 := x          -- FIM DO CALCULO DE Y1

f1 := f(t[1] , Y1)  -- calculo de f(t[1] , Y1)

out ! Y1          -- envio de Y1 para o Procedimento P2

in ? Y2           -- recebimento de Y2 do Procedimento P2

x := Y2

f2 := f(t[2] , x)   -- calculo de f(t[2] , Y2)

--CALCULO DE Y3

const1 := y - (((h*CIN)/DOIS)*(f1 - f2))
x := y
mo.dif := 10.0(REAL64)

-- METODO DE NEWTON PARA OBTER Y3

WHILE mo.dif > eps
  SEQ
    xaux := x
    PAR
      A := x - (((h*f(t[3] , x))/DOIS) + const1)
      Ax := UM - ((h/DOIS)*df(t[3] , x))
    x := x - (A/Ax)
    dif := x - xaux
    mo.dif := DSQRT((dif*dif))

Y3 := x          -- FIM DO CALCULO DE Y3

f3 := f(t[3] , Y3)  -- calculo de f(Y3)

in ? Y4           -- recebimento de Y4 do Procedimento P2

x := Y4

f4 := f(t[4] , x)   -- calculo de f(t[4] , Y4)

-- CALCULO DE y(t0 + jh)

y := y + (h*(((TR/DOIS)*(f2 + f4)) - (f1 + f3)))

out ! y          -- envio de y(t0 +jh) para o Procedimento P2

indj := indj + UM
tn := tn + h

ch ! y           --envio da solucao do problema de EDO para
                  -- o Procedimento Write
                  -- (este valor aparece na tela)

:
}}}

```

```

C-C- PROC P2
PROC P2 ( CHAN OF REAL64 in , out , VAL REAL64 t0 , T , y0 , eps)
  VAL UM IS 1.0 (REAL64) :
  VAL DOIS IS 2.0 (REAL64) :
  VAL TR IS 3.0 (REAL64) :
  VAL QT IS 4.0 (REAL64) :
  VAL CIN IS 5.0 (REAL64) :
  VAL DEZ IS 10.0 (REAL64) :
  REAL64 Y1 , Y2 , Y3 , Y4 , f1 , f2 , tn , comp.int , h :
  REAL64 x , xaux , mo.dif , ct2 , indj , y , dif , A , Ax :

  [5]REAL64 c :      --VETOR DE NOS DO METODO DE RK DE 4 ESTAGIOS (TIPO II)

  [5]REAL64 t :      --VETOR QUE ARMAZENA t      + h*c[i] (n-esimo passo)
                      -- n-1

  --CALCULO DE FUNCOES

  REAL64 FUNCTION DALOG ( VAL REAL64 X )      --FUNCAO LOGARITMO NEPERIANO
    REAL64 LN :
    VALOF
      LN := DALOG (X)
    RESULT LN
  :

  REAL64 FUNCTION DEXP (VAL REAL64 X)      --FUNCAO EXPONENCIAL
    REAL64 EXPON :
    VALOF
      EXPON := DEXP(X)
    RESULT EXPON
  :

  REAL64 FUNCTION f ( VAL REAL64 t , y )      --FUNCAO DO PVI  dy/dt = f(t,y)
                                              --y(t0) = y0
    VAL DEZ IS 10.0 (REAL64) :
    REAL64 FUN , ARG :
    VALOF
      SEQ
        ARG := DEZ*(t - y)
        FUN := DEXP(ARG)
      RESULT FUN
  :

  REAL64 FUNCTION df ( VAL REAL64 t , y )      --DERIVADA DA FUNCAO f
    VAL DEZ IS 10.0 (REAL64) :
    REAL64 Df , ARG :
    VALOF
      SEQ
        ARG := DEZ*(t - y)
        Df := -(DEZ*(DEXP(ARG)))
      RESULT Df
  :

  REAL64 FUNCTION DSQRT ( VAL REAL64 X )      --FUNCAO RAIZ QUADRADA
    REAL64 RAIZ.QUADRADA :
    VALOF
      RAIZ.QUADRADA := DSQRT(X)
    RESULT RAIZ.QUADRADA
  :

```

```

SEQ
  PAR
    c[1] := UM/DOIS
    c[2] := DOIS/TR
    c[3] := UM/DOIS
    c[4] := UM/TR

  out ? h      --RECEBIMENTO DO TAMANHO DO PASSO DO PROCEDIMENTO P1

  y := y0      -- VALOR INICIAL DO PVI

  tn := t0     -- EXTREMO INFERIOR DO INTERVALO [t0,T]

  comp.int := T - t0 --COMPRIMENTO DO INTERVALO [t0,T]

  indJ := 1.0(REAL64)

  WHILE (indj*h) <= (comp.int + eps)
    SEQ
      PAR i = 1 FOR 4
        t[i] := tn + (h*c[i])

        --CALCULO DE Y2 PELO METODO DE NEWTON

        x := y
        mo.dif := 10.0(REAL64)

        --METODO DE NEWTON

        WHILE mo.dif > eps
          SEQ
            xaux := x
            PAR
              A := x - (((DOIS*h)*f(t[2] , x))/TR) + y
              Ax := UM - (((DOIS*h)/TR)*df(t[2] , x))
            x := x - (A/Ax)
            dif := x - xaux
            mo.dif := DSQRT((dif*dif))

        Y2 := x      -- FIM DO CALCULO DE Y2

        f2 := f(t[2] ,Y2)      -- CALCULO DE f(Y2)

        out ? Y1      -- RECEBIMENTO DE Y1 DO PROCEDIMENTO P1

        in ! Y2      -- ENVIO DE Y2 PARA O PROCEDIMENTO P1

        x := Y1
        f1 := f(t[1] , x)      -- CALCULO DE f(t[1] , Y1)

        -- CALCULO DE Y4 PELO METODO DE NEWTON

        ct2 := y - ((h/TR)*((CIN*f1) - (QT*f2)))
        x := y + (UM/CIN)
        mo.dif := 10.0(REAL64)

        -- METODO DE NEWTON

```

```
WHILE mo.dif > eps
```

```
SEQ
```

```
    xaux := x
```

```
    PAR
```

```
        A := x - (ct2 + (((DOIS*h)*f(t[4] , x))/TR))
```

```
        Ax := UM - (((h*DOIS)/TR)*df(t[4] , x))
```

```
    x := x - (A/Ax)
```

```
    dif := x - xaux
```

```
    mo.dif := DSQRT((dif*dif))
```

```
Y4 := x          -- FIM DO CALCULO DE Y4
```

```
in ! Y4          -- ENVIO DE Y4 PARA O PROCESSADOR 1 (P1)
```

```
out ? y          -- RECEBIMENTO DE y(t0 + jh) DO PROCEDIMENTO P1
```

```
tn := tn + h
```

```
indj := indj + UM
```

```
:
```

```
}}}
```

```
{{{ PROC WRITE
```

```
PROC WRITE (CHAN OF INT keyboard , CHAN OF ANY screen , CHAN OF REAL64 ch )
```

```
    REAL64 y , h :
```

```
    SEQ
```

```
    ch ? h  --RECEBIMENTO DO TAMANHO DE PASSO DO PROCEDIMENTO P1
```

```
    newline (screen)
```

```
    newline (screen)
```

```
    write.full.string (screen, " PARA h = ")
```

```
    write.real64 (screen , h , 2 , 06)
```

```
    newline (screen)
```

```
    newline (screen)
```

```
    write.full.string (screen , "A SOLUCAO DO PVI E DADA POR" )
```

```
    ch ? y  --RECEBENDO DO PROCEDIMENTO P1 A SOLUCAO DO PVI
```

```
    write.real64 (screen , y , 2 , 10 )
```

```
    newline (screen)
```

```
    newline (screen)
```

```
:
```

```
}}}
```

```
--PROGRAMA PRINCIPAL
```

```
--DECLARACOES DE CANAIS
```

```
CHAN OF REAL64 in , out , ch :
```

```
--DECLARACOES DE VARIAVEIS
```

```
VAL N10 IS 10.0 (REAL64) :
```

```
INT kchar :
```

```
REAL64 h :
```

```
--VARIAVEL QUE CONTROLA O TAMANHO DO PASSO
```

```
--NO METODO DE RK .
```

```
SEQ
```

```
h := 0.5 (REAL64)
```

```
--TAMANHO DO PASSO INICIAL
```

```
WHILE h >= 0.001(REAL64)
```

```
SEQ
```

```
PAR
```

```
PAR
```

```
    P1 ( in , out , ch , t0 , T , y0 , eps , h)
```

```
    P2 ( in , out , t0 , T , y0 , eps )
```

```
    WRITE ( keyboard , screen , ch )
```

```
h := h/N10
```

MÉTODO PDIRK

```

#USE userio
#USE uservals
#USE interf
#USE dblmath
{{{ PROC P1
PROC P1 (CHAN OF REAL64 in , out , ch , VAL REAL64 t0 , T , eps ,
        VAL [JREAL64 y0 , REAL64 h )
    VAL n.1 IS 1.0(REAL64) :
    VAL n.5 IS 5.0(REAL64) :
    VAL n.10 IS 10.0(REAL64) :
    VAL n.12 IS 12.0(REAL64) :
    REAL64 F , G , d1F , d2F , d1G , d2G , cnt1 , cnt2 , d :
    REAL64 cont , mod.dif , dif , ex , ey , det , comp.int :

    [JREAL64 A1 : --PRIMEIRA LINHA DA MATRIZ DO MET. 2-RK RADAU IIA

    [JREAL64 Y1 , Y2 , f1 , f2 , x , y ,iaux :

    --DECLARAÇÕES DE FUNÇÕES

    {{{ FUNÇÕES
    REAL64 FUNCTION DSQRT (VAL REAL64 X)
        REAL64 RAIZ.0 :
        VALOF
            RAIZ.0 := DSQRT(X)
        RESULT RAIZ.0
    :

    REAL64 FUNCTION f (VAL INT 1 , VAL [JREAL64 y ) --FUNÇÃO DO SISTEMA DE EDO
        VAL n.24 IS 24.0(REAL64) : --DE DIMENSÃO DOIS
        VAL n.25 IS 25.0(REAL64) : --y' = f(y(t))
        VAL n.26 IS 26.0(REAL64) : --y(0) = y(0)
        REAL64 FUN :
        VALOF
            SEQ
            IF
                1 = 1
                FUN := (n.25*y[2]) - (n.24*y[1])
                1 = 2
                FUN := (n.26*y[1]) - (n.25*y[2])
            TRUE
            SKIP
        RESULT FUN

```

```

REAL64 FUNCTION df ( VAL INT 1 , m , VAL CREAL64 y) --JACOBIANO DA
REAL64 Df :
VAL n.24 IS 24.0(REAL64) :
VAL n.25 IS 25.0(REAL64) :
VAL n.26 IS 26.0(REAL64) :
VALOF
  SEQ
    IF
      1 = 1
      IF
        m = 1
          Df := - n.24
        m = 2
          Df := n.25
      TRUE
      SKIP
    1 = 2
    IF
      m = 1
        Df := n.26
      m = 2
        Df := - n.25
    TRUE
    SKIP
  RESULT Df
:
}}

```

SEQ

```

  PAR
    A1[1] := n.5/n.12
    A1[2] := -(n.1/n.12)
    y[1] := y0[1]
    y[2] := y0[2]

```

--ENVIO DO TAMANHO DO PASSO PARA OS PROCESSADORES P2 E WRITE

```

  PAR
    out ! h
    ch ! h

```

```

  d := 0.3025345782(REAL64)
  cont := n.1
  comp.int := T - t0
  comp.int := comp.int + eps
  out ! comp.int

```

WHILE (cont*h) <= comp.int --CONTROLE DE PASSOS DO 2-RK

```

  SEQ
    PAR i = 1 FOR 2
      x[i] := y[i]
    mod.dif := 10.0(REAL64)

```

--0

--METODO DE NEWTON PARA OBTER O VETOR BI-DIMENSIONAL Y1

--CORRESPONDENTES AO METODO DE 2-PDIRK


```

WHILE mod.dif > eps
  SEQ
    F := (x[1]-y[1]) - ((h*d)*f(1,x))
    G := (x[2]-y[2]) - ((h*d)*f(2,x))
    d1F := n.1 - ((h*d)*df(1,1,x))
    d2F := - ((h*d)*df(1,2,x))
    d1G := - ((h*d)*df(2,1,x))
    d2G := n.1 - ((h*d)*df(2,2,x))
    det := (d1F*d2G) - (d2F*d1G)
    PAR
      ex := ((G*d2F) - (F*d2G))/det
      ey := ((F*d1G) - (G*d1F))/det
    PAR
      x[1] := x[1] + ex
      x[2] := x[2] + ey
      mod.dif := (ex*ex) + (ey*ey)
      mod.dif := DSQRT(mod.dif)
      --0
      --VALORES DAS COORDENADAS DO VETOR Y1

  PAR i = 1 FOR 2
    Y1[i] := x[i]
    --0
    --CALCULO DE f1(Y1) E f2(Y1)

  PAR
    f1[1] := f(1,Y1)
    f2[1] := f(2,Y1)
    --0
    --ENVIO DO VETOR Y1 PARA O PROCESSADOR 1

  SEQ i = 1 FOR 2
    out ! Y1[i]
    --0
    --RECEBENDO DO PROCESSADOR 2 O VETOR Y2

  SEQ i = 1 FOR 2
    in ? Y2[i]

  PAR i = 1 FOR 2
    x[i] := Y2[i]
    --0
    --CALCULO DE f1(Y2) E f2(Y2)

  PAR
    f1[2] := f(1,x)
    f2[2] := f(2,x)

    --INICIO DAS ITERACOES REFERENTES AO METODO ITERATIVO DIAGONAL
    --PDIRK

```

```

SEQ k = 1 FOR 3
  SEQ
    PAR
      x[1] := -(A1[1]*f1[1])
      x[2] := -(A1[1]*f2[1])
      xaux[1] := -(A1[2]*f1[2])
      xaux[2] := -(A1[2]*f2[2])
      cnt1 := d*f1[1]
      cnt2 := d*f2[1]
    PAR
      cnt1 := (h*(x[1]+(xaux[1]+cnt1))) - y[1]
      cnt2 := (h*(x[2]+(xaux[2]+cnt2))) - y[2]
    PAR i = 1 FOR 2
      x[i] := Y1[i]
    mod.dif := 10.0(REAL64)
  --k
  --METODO DE NEWTON PARA OBTER Y1

  WHILE mod.dif > eps
    SEQ
      F := (x[1]+cnt1) - ((h*d)*f(1,x))
      G := (x[2]+cnt2) - ((h*d)*f(2,x))
      d1F := n.1 - ((h*d)*df(1,1,x))
      d2F := - ((h*d)*df(1,2,x))
      d1G := - ((h*d)*df(2,1,x))
      d2G := n.1 - ((h*d)*df(2,2,x))
      det := (d1F*d2G) - (d2F*d1G)
      PAR
        ex := ((G*d2F) - (F*d2G))/det
        ey := ((F*d1G) - (G*d1F))/det
      PAR
        x[1] := x[1] + ex
        x[2] := x[2] + ey
      mod.dif := (ex*ex) + (ey*ey)
      mod.dif := DSQRT(mod.dif)
    --k
    --VALOR DAS COMPONENTES DO VETOR Y1

    PAR i = 1 FOR 2
      Y1[i] := x[i]
    --k
    --CALCULO DE f1(Y1) E f2(Y1)

    PAR
      f1[1] := f(1,Y1)
      f2[1] := f(2,Y1)
    --k
    --ENVIO DO VETOR Y1 PARA O PROCESSADOR 2

    SEQ i = 1 FOR 2
      out ! Y1[i]
    --k
    --RECEBENDO O VETOR Y2 DO PROCESSADOR 2

    SEQ i = 1 FOR 2
      in ? Y2[i]
    PAR i = 1 FOR 2
      x[i] := Y2[i]

```

```

--k      --k
--CALCULO DE f1(Y2 ) E f2(Y2 )

```

```

PAR
  f1[2] := f(1,x)
  f2[2] := f(2,x)

```

```

--ATUALIZACAO DO VETOR y QUE DARA A SOLUCAO DO PVI

```

```

PAR i = 1 FOR 2
  y[i] := Y2[i]
  cont := cont + n.1

```

```

--ENVIO DO VETOR SOLUCAD PARA O PROCESSADOR WRITE

```

```

SEQ i = 1 FOR 2
  ch ! y[i]

```

```

:
}}}
{{{ PROC P2
PROC P2 ( CHAN OF REAL64 in , out , VAL REAL64 eps , VAL [JREAL64 y0 )

```

```

  VAL n.1 IS 1.0(REAL64) :
  VAL n.3 IS 3.0(REAL64) :
  VAL n.4 IS 4.0(REAL64) :
  VAL n.10 IS 10.0(REAL64) :
  REAL64 h , F , G , d1F , d2F , d1G , d2G , cnt1 , cnt2 , d , det :
  REAL64 cont , mod.dif , dif , xaux , ex , ey , comp.int :

```

```

[3]REAL64 A2 : --SEGUNDA LINHA DA MATRIZ DO MET. 2-RK RADAU IIA

```

```

[3]REAL64 Y1 , Y2 , f1 , f2 , x , xaux , y :

```

```

{{{ FUNCOES
REAL64 FUNCTION DSQRT (VAL REAL64 X)
  REAL64 RAIZ.0 :
  VALOF
    RAIZ.0 := DSQRT(X)
  RESULT RAIZ.0

```

```

:
REAL64 FUNCTION f ( VAL INT 1 , VAL [JREAL64 y) --FUNCAO f DO SISTEMA
  REAL64 FUN : --y' = f(y(t)) = (f1,f2)
  VAL n.24 IS 24.0(REAL64) : --y(0) = y0
  VAL n.25 IS 25.0(REAL64) :
  VAL n.26 IS 26.0(REAL64) :
  VALOF
    SEQ
      IF
        1 = 1
          FUN := (n.25*y[2]) - (n.24*y[1])
        1 = 2
          FUN := (n.26*y[1]) - (n.25*y[2])
      TRUE
      SKIP
    RESULT FUN

```

```

REAL64 FUNCTION df (VAL INT 1 , m , VAL [JREAL64 y) --JACOBIANO DA
REAL64 Df :
VAL n.24 IS 24.0(REAL64) :
VAL n.25 IS 25.0(REAL64) :
VAL n.26 IS 26.0(REAL64) :
VALDF
SEQ
  IF
    1 = 1
    IF
      m = 1
      Df := - n.24
      m = 2
      Df := n.25
      TRUE
      SKIP
    1 = 2
    IF
      m = 1
      Df := n.26
      m = 2
      Df := - n.25
      TRUE
      SKIP
  RESULT Df
:
}}}
SEQ
  PAR
    A2[1] := n.3/n.4
    A2[2] := n.1/n.4
    y[1] := y0[1]
    y[2] := y0[2]

out ? h --RECEBENDO DO PROCESSADOR 1 O TAMANHO DO PASSO

d := 0.3025345782(REAL64)
cont := n.1
out ? comp.int

WHILE (cont*h) <= comp.int --CONTOLE DOS PASSOS DO METODO 2-RK
SEQ
  PAR i = 1 FOR 2
  RADAU IIA
  x[i] := y[i]
  mod.dif := 10.0(REAL64)
  --0
  --METODO DE NEWTON PARA OBTEN Y2 CORRESPONDENTE AO METODO
  --DIAGONAL PDIRK

```

```

WHILE mod.dif > eps
  SEQ
    F := (x[1]-y[1]) - ((h*d)*f(1,x))
    G := (x[2]-y[2]) - ((h*d)*f(2,x))
    d1F := n.1 - ((h*d)*df(1,1,x))
    d2F := - ((h*d)*df(1,2,x))
    d1G := - ((h*d)*df(2,1,x))
    d2G := n.1 - ((h*d)*df(2,2,x))
    det := (d1F*d2G) - (d2F*d1G)
    PAR
      ex := ((G*d2F) - (F*d2G))/det
      ey := ((F*d1G) - (G*d1F))/det
    PAR
      x[1] := x[1] + ex
      x[2] := x[2] + ey
      mod.dif := (ex*ex) + (ey*ey)
      mod.dif := DSQRT(mod.dif)
  --0
  --VALORES DAS COORDENADAS DO VETOR Y2
  PAR i = 1 FOR 2
    Y2[i] := x[i]
  --0
  --CALCULO DE f1(Y2) E f2(Y2)
  PAR
    f1[2] := f(1,Y2)
    f2[2] := f(2,Y2)
  --0
  --RECEBENDO Y1 DO PROCESSADOR 1
  SEQ i = 1 FOR 2
    out ? Y1[i]
  --0
  --ENVIO DO VETOR Y2 PARA O PROCESSADOR 1
  SEQ i = 1 FOR 2
    in ! Y2[i]
  PAR i = 1 FOR 2
    x[i] := Y1[i]
  --0
  --CALCULO DE f1(Y1) E f2(Y1)
  PAR
    f1[1] := f(1,x)
    f2[1] := f(2,x)
  --INICIO DAS ITERACOES DO METODO 2-PDIRK
  SEQ k = 1 FOR 3
    SEQ
      PAR
        x[1] := -(A2[1]*f1[1])
        x[2] := -(A2[1]*f2[1])
        xaux[1] := -(A2[2]*f1[2])
        xaux[2] := -(A2[2]*f2[2])
        cnt1 := d*f1[2]
        cnt2 := d*f2[2]
        cnt1 := (h*(x[1]+(xaux[1]+cnt1))) - y[1]
        cnt2 := (h*(x[2]+(xaux[2]+cnt2))) - y[2]
      PAR i = 1 FOR 2
        x[i] := Y2[i]
      mod.dif := 10.0(REAL64)

```

```

--METODO DE NEWTON PARA OBTEN O VETOR Y2
WHILE mod.dif > 0.000000001 (REAL64)
  SEQ
    F := (x[1]+cnt1) - ((h*d)*f(1,x))
    G := (x[2]+cnt2) - ((h*d)*f(2,x))
    d1F := n.1 - ((h*d)*df(1,1,x))
    d2F := - ((h*d)*df(1,2,x))
    d1G := - ((h*d)*df(2,1,x))
    d2G := n.1 - ((h*d)*df(2,2,x))
    det := (d1F*d2G) - (d2F*d1G)
    PAR
      ex := ((G*d2F) - (F*d2G))/det
      ey := ((F*d1G) - (G*d1F))/det
    PAR
      x[1] := x[1] + ex
      x[2] := x[2] + ey
      mod.dif := (ex*ex) + (ey*ey)
      mod.dif := DSORT(mod.dif)
  --k
  --VALORES DAS COORDENADAS DO VETOR Y2

  PAR i = 1 FOR 2
    Y2[i] := x[i]
  --k
  --CALCULO DE f1(Y2) E f2(Y2)

  PAR
    f1[2] := f(1,Y2)
    f2[2] := f(2,Y2)
  --k
  --RECEBENDO Y1 DO PROCESSADOR 1

  SEQ i = 1 FOR 2
    out ? Y1[i]
  --k
  --ENVIANDO Y2 PARA O PROCESSADOR 1

  SEQ i = 1 FOR 2
    in ! Y2[i]
  PAR i = 1 FOR 2
    x[i] := Y1[i]
  --k
  --CALCULO DE f1(Y1) E f2(Y1)

  PAR
    f1[1] := f(1,x)
    f2[1] := f(2,x)
  --ATUALIZACAO DO VETOR y QUE DARA A SOLUCAO DO PVI

  PAR i = 1 FOR 2
    y[i] := Y2[i]
  cont := cont + n.1

```

```

{{{ PROC WRITE
PROC WRITE ( CHAN OF INT keyboard , CHAN OF ANY screen , CHAN OF REAL64 ch)
  REAL64 h :
  [3]REAL64 y :
  VAL n.10 IS 10.0(REAL64) :
  SEQ

  ch ? h --RECEBENDO DO PROCESSADOR 1 O TAMANHO DO PASSO

  write.full.string (screen , "PARA h = ")
  write.real64 (screen , h , 2 , 06)
  newline (screen)
  newline (screen)
  write.full.string ( screen , " A SOLUCAO DO PVI E DADA POR ")
  newline (screen)
  newline (screen)
  SEQ i = 1 FOR 2
    SEQ
      ch ? y[i]
      write.full.string (screen, "componente.")
      write.int(screen , i , 1)
      write.full.string ( screen , " = " )
      write.real64 (screen , y[i] , 2 , 10 )
      newline (screen)
      newline (screen)
    :
  }}}
--PROGRAMA PRINCIPAL
CHAN OF REAL64 in , out , ch :
INT kchar :
VAL n.10 IS 10.0(REAL64) :
REAL64 h :

--EXTREMOS DO INTERVALO DE INTEGRACAO [t0,T]

VAL t0 IS 0.0(REAL64) :
VAL T IS 1.0(REAL64) : --VAMOS ENCONTRAR A APROXIMACAO DE y(T)

VAL eps IS 0.00000000001(REAL64) :

[3]REAL64 y0 : --VALOR INICIAL DO PVI

SEQ
  h := 0.5(REAL64)
  PAR
    y0[1] := 1.0025(REAL64)
    y0[2] := 0.9974(REAL64)
  WHILE h >= 0.001(REAL64)
    SEQ
      PAR
        PAR
          P1 ( in , out , ch , t0 , T , eps , y0 , h )
          P2 ( in , out , eps , y0 )
        WRITE ( keyboard , screen , ch )
      h := h/n.10
    write.full.string ( screen , " TERMINADO O PROCESSO PRESSIONE UMA TECLA")
    read.char (keyboard , kchar)

```

METODO PDIRK ATÉ CONVERGÊNCIA

```
#USE userio.
#USE uservals
#USE interf
#USE dblmath
VAL eps IS 0.0000000001(REAL64) :  --TOLERANCIA DO ERRO

VAL t0 IS 0.0(REAL64) :  --EXTREMO INFERIOR DO INTERVALO [t0,T] QUE SERA
                        --DISCRETIZADO EM PONTOS IGUALMENTE ESPACADOS

VAL T IS 1.0(REAL64) :  --EXTREMO SUPERIOR DO INTERVALO [t0,T].
                        --VAMOS CALCULAR y(T) APLICANDO METODO PDIRK ATE
                        --CONVERGENCIA , 2-ESTAGIOS E 2-PROCESSADORES , AO
                        --PVI dy/dt = f(t,y) , y(t0) = y0
```

```
{{{ PROCEDIMENTO 1
```

```
PROC P1 (CHAN OF REAL64 in , out , ch , VAL REAL64 t0 , T , eps ,
        VAL [J]REAL64 y0 , REAL64 h)
```

```
{{{ variaveis e funcoes
```

```
VAL n.1 IS 1.0(REAL64) :
```

```
VAL n.2 IS 2.0(REAL64) :
```

```
VAL n.3 IS 3.0(REAL64) :
```

```
VAL n.4 IS 4.0(REAL64) :
```

```
VAL n.81 IS 81.0(REAL64) :
```

```
VAL n.216 IS 216.0(REAL64) :
```

```
VAL n.288 IS 288.0(REAL64) :
```

```
REAL64 cont , comp.int , soma , tn , arg :
```

```
REAL64 dif , mod.dif , difY , mod.difY , det , ex , ey :
```

```
[J]REAL64 f1 , f2 , S1 , S2 :
```

```
[J]REAL64 x , dF , dG , cnt :
```

```
[J]REAL64 y :  --VARIABEL QUE VAI ARMazenAR A SOLUCAO DO PVI
```

```
REAL64 F , G :  --VARIABEIS UTILIZADAS NO METODO DE NEWTON
```

```
REAL64 a1 :  -- PRIMEIRA COMPONENTE DO VETOR DE PARAMETROS ASSOCIADO
            -- AO METODO DE LAGRANGE
```

```
REAL64 d1 :  -- ELEMENTO DIAGONAL DA MATRIZ DE ITERACAO
```

```
[J]REAL64 A1 :  -- PRIMEIRA LINHA DA MATRIZ DO METODO DE LAGRANGE
```

```
[J]REAL64 Y1 , Y2 :  --APROXIMACOES INTERNAS DO METODO 2-RK (LAGRANGE)
```

```
[J]REAL64 Yold1 , Yold2 :  --AUXILIARES DAS APROXIMACOES INTERNAS
```

```
[J]REAL64 c :  --VETOR DE NOS DO METODO 2-Rk (LAGRANGE)
```

```
[J]REAL64 t :  --VETOR QUE CALCULA t + h*c[i] (N-ESIMO PASSO)
            -- n-1
```



```

REAL64 FUNCTION DSWRT (VAL REAL64 X)
  REAL64 RAIZ.QUADRADA :
  VALOF
    RAIZ.QUADRADA := DSWRT(X)
  RESULT RAIZ.QUADRADA
:

REAL64 FUNCTION DEXP (VAL REAL64 X)
  REAL64 EXPON :
  VALOF
    EXPON := DEXP(X)
  RESULT EXPON
:

REAL64 FUNCTION f (VAL INT 1 , VAL [REAL64 y) --FUNCAO DO SISTEMA
                                           --DE DIMENSAO DOIS
                                           --y' = f(y(t))
                                           --y(0) = y0

  REAL64 FUN :
  VAL n.24 IS 24.0(REAL64) :
  VAL n.25 IS 25.0(REAL64) :
  VAL n.26 IS 26.0(REAL64) :
  VALOF
    SEQ
    IF
      1 = 1
      FUN := (n.25*y[2]) - (n.24*y[1])
      1 = 2
      FUN := (n.26*y[1]) - (n.25*y[2])
    TRUE
    SKIP
  RESULT FUN
:

REAL64 FUNCTION df ( VAL INT 1 , m , VAL [REAL64 y ) --JACOBIANO DA
                                           FUNCAO f

  REAL64 Df , ARG :
  VAL n.24 IS 24.0(REAL64) :
  VAL n.25 IS 25.0(REAL64) :
  VAL n.26 IS 26.0(REAL64) :
  VALOF
    SEQ
    IF
      1 = 1
      IF
        m = 1
        Df := - n.24
        m = 2
        Df := n.25
      TRUE
      SKIP
      1 = 2
      IF
        m = 1
        Df := n.26
        m = 2
        Df := - n.25
      TRUE
      SKIP
    TRUE
    SKIP
  RESULT Df
:
}}}

```

```

SEQ
  arg := n.4*(DSQRT(n.2) + n.1)
  PAR
    c[1] := n.3/n.4
    c[2] := n.1
    A1[1] := n.216/n.288
    A1[2] := -(n.81/n.288)
    d1 := n.3/arg
    a1 := n.81/n.288

  PAR
    out ! h    --ENVIO DO TAMANHO DO PASSO PARA O PROCEDIMENTO P2
    ch ! h     --ENVIO DO TAMANHO DO PASSO PARA O PROCEDIMENTO WRITE

  PAR 1 = 1 FOR 2
    y[1] := y0[1]
  tn := t0
  comp.int := T - t0
  cont := 1.0(REAL64)

  WHILE (cont*h) <= (comp.int + eps) --CONTROLE DE PASSOS DO 2-RK
    --LAGRANGE
    SEQ
      mod.difY := 10.0(REAL64)
      PAR i = 1 FOR 2
        t[i] := tn + (h*c[i])
      PAR
        SEQ 1 = 1 FOR 2
          Y1[1] := y[1]
        SEQ 1 = 1 FOR 2
          Y2[1] := y[1]
        SEQ 1 = 1 FOR 2
          Yold1[1] := y[1]
        SEQ 1 = 1 FOR 2
          Yold2[1] := y[1]
      PAR
        SEQ 1 = 1 FOR 2
          f1[1] := f(1,Y1)
        SEQ 1 = 1 FOR 2
          f2[1] := f(1,Y2)

    --ITERACOES DO METODO PDIRK ATE CONVERGENCIA

    WHILE mod.difY > eps
      SEQ
        PAR
          SEQ 1 = 1 FOR 2
            S1[1] := -(A1[1]*f1[1])
          SEQ 1 = 1 FOR 2
            S2[1] := -(A1[2]*f2[1])
          SEQ 1 = 1 FOR 2
            cnt[1] := d1*f1[1]
          SEQ 1 = 1 FOR 2
            x[1] := -(a1*f(1 , y))
        SEQ 1 = 1 FOR 2
          cnt[1] := (h*(((S1[1]+S2[1])+cnt[1])+x[1])) - y[1]
        mod.dif := 10.0(REAL64)
        PAR 1 = 1 FOR 2
          x[1] := Y1[1]

```

```

--METODO DE NEWTON PARA OBTEN O VETOR BI-DIMENSIONAL Y1 --k
--CORRESPONDENTE AO METODO PDIRK ATE CONVERGENCIA

WHILE mod.dif > eps
  SEQ
    PAR
      F := (x[1] + cnt[1]) - ((h*d1)*f(1,x))
      G := (x[2] + cnt[2]) - ((h*d1)*f(2,x))
      dF[1] := n.1 - ((h*d1)*df(1,1,x))
      dF[2] := - ((h*d1)*df(1,2,x))
      dG[1] := - ((h*d1)*df(2,1,x))
      dG[2] := n.1 - ((h*d1)*df(2,2,x))
      det := (dF[1]*dG[2]) - (dF[2]*dG[1])
      PAR
        ex := ((G*dF[2]) - (F*dG[2]))/det
        ey := ((F*dG[1]) - (G*dF[1]))/det
      PAR
        x[1] := x[1] + ex
        x[2] := x[2] + ey
        mod.dif := (ex*ex) + (ey*ey)
        mod.dif := DSQRT(mod.dif)
    --k
  --VALORES DAS COORDENADAS DE Y1

  PAR 1 = 1 FOR 2
    Y1[1] := x[1]
    --k
  --CALCULO DE f(Y1 )

  PAR 1 = 1 FOR 2
    f1[1] := f(1 , Y1)
    --k
  --ENVIO DO VETOR Y1 PARA O PROCEDIMENTO P2

  SEQ 1 = 1 FOR 2
    out ! Y1[1]
    --k
  --RECEBENDO DO PROCEDIMENTO P2 O VETOR Y2

  SEQ 1 = 1 FOR 2
    in ? Y2[1]
    --k
  --CALCULO DE f(Y2 )

  PAR 1 = 1 FOR 2
    x [1] := Y2[1]
  PAR 1 = 1 FOR 2
    f2[1] := f(1 , x)

```

```

    PAR
      SEQ 1 = 1 FOR 2
        S1[1] := Yold1[1] - Y1[1]
      SEQ 1 = 1 FOR 2
        S2[1] := Yold2[1] - x[1]
      SEQ 1 = 1 FOR 2
        difY := ((S1[1]*S1[1]) + (S2[1]*S2[1])) + difY
      mod.difY := DSQRT(difY)
      PAR 1 = 1 FOR 2
        PAR
          Yold1[1] := Y1[1]
          Yold2[1] := x[1]

--ATUALIZACAO DO VETOR y NO FINAL DE UM PASSO

    PAR 1 = 1 FOR 2
      y[1] := Y2[1]

    tn := tn + h
    cont := cont + n.1

--ENVIO DO VETOR SOLUCAO PARA O PROCEDIMENTO WRITE

```

```

    SEQ 1 = 1 FOR 2
      ch ! y[1]
    :
  }}}
  {{{ PROCEDIMENTO 2
  PROC P2 (CHAN OF REAL64 in , out , VAL REAL64 t0 , T , eps ,
    VAL [3]REAL64 y0 )

```

```

  {{{ variaveis e funcoes

```

```

  VAL n.1 IS 1.0(REAL64) :
  VAL n.2 IS 2.0(REAL64) :
  VAL n.3 IS 3.0(REAL64) :
  VAL n.4 IS 4.0(REAL64) :
  VAL n.6 IS 6.0(REAL64) :
  VAL n.10 IS 10.0(REAL64) :
  VAL n.48 IS 48.0(REAL64) :
  VAL n.80 IS 80.0(REAL64) :
  VAL n.256 IS 256.0(REAL64) :
  VAL n.288 IS 288.0(REAL64) :

```

```

  REAL64 cont , comp.int , soma , tn , arg , h , ex , ey , det :

```

```

  REAL64 dif , mod.dif , difY , mod.difY , F , G :
  [3]REAL64 dF , dG , x , f1 , f2 , S1 , S2 , cnt :
  [3]REAL64 Y1 , Y2 , y :
  [3]REAL64 Yold1 , Yold2 :
  [3]REAL64 c :
  [3]REAL64 t :
  [3]REAL64 A2 :
  REAL64 a2 :
  REAL64 d2 :

```

```

  REAL64 FUNCTION DSQRT (VAL REAL64 X)
    REAL64 RAIZ.QUADRADA :
    VALOF
      RAIZ.QUADRADA := DSQRT(X)
    RESULT RAIZ.QUADRADA

```

```

  :

```

```
REAL64 FUNCTION DEXP (VAL REAL64 X)
```

```
REAL64 EXPON :
```

```
VALOF
```

```
EXPON := DEXP(X)
```

```
RESULT EXPON
```

```
:
```

```
REAL64 FUNCTION f (VAL INT 1 , VAL [REAL64 y) --FUNCAO DO SISTEMA
```

```
--DE DIMENSAO DOIS
```

```
--y' = f(y(t))
```

```
--y(0) = y0
```

```
REAL64 FUN , ARG :
```

```
VAL n.24 IS 24.0(REAL64) :
```

```
VAL n.25 IS 25.0(REAL64) :
```

```
VAL n.26 IS 26.0(REAL64) :
```

```
VALOF
```

```
SEQ
```

```
IF
```

```
1 = 1
```

```
FUN := (n.25*y[2]) - (n.24*y[1])
```

```
1 = 2
```

```
FUN := (n.26*y[1]) - (n.25*y[2])
```

```
TRUE
```

```
SKIP
```

```
RESULT FUN
```

```
:
```

```
REAL64 FUNCTION df (VAL INT 1 , m , VAL [REAL64 y) --JACOBIANO DA
```

```
FUNCAO f
```

```
REAL64 Df , ARG :
```

```
VAL n.24 IS 24.0(REAL64) :
```

```
VAL n.25 IS 25.0(REAL64) :
```

```
VAL n.26 IS 26.0(REAL64) :
```

```
VALOF
```

```
SEQ
```

```
IF
```

```
1 = 1
```

```
IF
```

```
m = 1
```

```
Df := - n.24
```

```
m = 2
```

```
Df := n.25
```

```
TRUE
```

```
SKIP
```

```
1 = 2
```

```
IF
```

```
m = 1
```

```
Df := n.26
```

```
m = 2
```

```
Df := - n.25
```

```
TRUE
```

```
SKIP
```

```
TRUE
```

```
SKIP
```

```
RESULT Df
```

```
:
```

```
}}}
```

```

SEQ
  arg := n.6*(DSQRT(n.2) - n.1)
  PAR
    c[1] := n.3/n.4
    c[2] := n.1
    a2 := n.80/n.288
    A2[1] := n.256/n.288
    A2[2] := -(n.48/n.288)
    d2 := n.1/arg

  out ? h  --RECEBENDO DO PROCEDIMENTO P1 O TAMANHO DO PASSO

  PAR 1 = 1 FOR 2
    y[1] := y0[1]
  tn := t0
  comp.int := T - t0
  cont := 1.0(REAL64)

  WHILE (cont*h) <= (comp.int + eps)  --CONTROLE DE PASSOS DO 2-RK
    --LAGRANGE
    SEQ
      mod.difY := 10.0(REAL64)
      PAR i = 1 FOR 2
        t[i] := tn + (h*c[i])
      PAR
        SEQ 1 = 1 FOR 2
          Y1[1] := y[1]
        SEQ 1 = 1 FOR 2
          Y2[1] := y[1]
        SEQ 1 = 1 FOR 2
          Yold1[1] := y[1]
        SEQ 1 = 1 FOR 2
          Yold2[1] := y[1]
      PAR
        SEQ 1 = 1 FOR 2
          f1[1] := f(1, Y1)
        SEQ 1 = 1 FOR 2
          f2[1] := f(1, Y2)

    --ITERACOES DO METODO PDIRK ATE CONVERGENCIA
    WHILE mod.difY > eps
      SEQ
        PAR
          SEQ 1 = 1 FOR 2
            S1[1] := -(A2[1]*f1[1])
          SEQ 1 = 1 FOR 2
            S2[1] := -(A2[2]*f2[1])
          SEQ 1 = 1 FOR 2
            cnt[1] := d2*f2[1]
          SEQ 1 = 1 FOR 2
            x[1] := -(a2*f(1, y))
        SEQ 1 = 1 FOR 2
          cnt[1] := (h*(((S1[1]+S2[1])+cnt[1])+x[1])) - y[1]
        mod.dif := n.10
        PAR 1 = 1 FOR 2
          x[1] := Y2[1]

```

```

--METODO DE NEWTON PARA OBTEN O VETOR BI-DIMENSIONAL Y2
--CORRESPONDENTE AO METODO POIRK ATE CONVERGENCIA
--k
WHILE mod.dif > eps
  SEQ
    PAR
      F := (x[1]+cnt[1]) - ((h*d2)*f(1 , x))
      G := (x[2]+cnt[2]) - ((h*d2)*f(2 , x))
      dF[1] := n.1 - ((h*d2)*df(1,1,x))
      dF[2] := - ((h*d2)*df(1,2,x))
      dG[1] := - ((h*d2)*df(2,1,x))
      dG[2] := n.1 - ((h*d2)*df(2,2,x))
      det := (dF[1]*dG[2]) - (dF[2]*dG[1])
    PAR
      ex := ((G*dF[2]) - (F*dG[2]))/det
      ey := ((F*dG[1]) - (G*dF[1]))/det
    PAR
      x[1] := x[1] + ex
      x[2] := x[2] + ey
      mod.dif := (ex*ex) + (ey*ey)
      mod.dif := DSQRT(mod.dif)
--k
--COORDENADAS DO VETOR Y2
PAR 1 = 1 FOR 2
  Y2[1] := x[1]
--k
--CALCULO DE f(Y2 )
PAR 1 = 1 FOR 2
  f2[1] := f(1 , Y2)
--k
--RECEBENDO DO PROCEDIMENTO P1 O VETOR Y1
SEQ 1 = 1 FOR 2
  out ? Y1[1]
--k
--ENVIO DO VETOR Y2 PARA O PROCEDIMENTO P1
SEQ 1 = 1 FOR 2
  in ! Y2[1]
--k
--CALCULO DE f(Y1 )
PAR 1 = 1 FOR 2
  x[1] := Y1[1]
PAR 1 = 1 FOR 2
  f1[1] := f(1 , x)

```

```

    difY := 0.0(REAL64)
    PAR
        SEQ 1 = 1 FOR 2
            S1[1] := Yold1[1] - x[1]
        SEQ 1 = 1 FOR 2
            S2[1] := Yold2[1] - Y2[1]
        SEQ 1 = 1 FOR 2
            difY := ((S1[1]*S1[1]) + (S2[1]*S2[1])) + difY
        mod.difY := DSQRT(difY)
    PAR
        SEQ 1 = 1 FOR 2
            Yold1[1] := x[1]
        SEQ 1 = 1 FOR 2
            Yold2[1] := Y2[1]

    --ATUALIZACAO DO VETOR y NO FINAL DE UM PASSO

    PAR 1 = 1 FOR 2
        y[1] := Y2[1]

    tn := tn + h
    cont := cont + n.1
:
}}
{{{ PROCEDIMENTO WRITE
PROC WRITE ( CHAN OF INT keyboard , CHAN OF ANY screen , CHAN OF REAL64 ch)
    REAL64 h :
    [3]REAL64 y :
    SEQ

    ch ? h --RECEBENDO DO PROCEDIMENTO P1 O TAMANHO DO PASSO

    newline(screen)
    newline(screen)
    write.full.string (screen , "Para h = ")
    write.real64 (screen , h , 2 , 05)
    newline(screen)
    newline(screen)
    write.full.string (screen , " A SOLUCAO DO PVI E DADA POR ")
    newline(screen)
    newline(screen)
    SEQ 1 = 1 FOR 2
        SEQ

    ch ? y[1] --RECEBENDO DO PROCEDIMENTO P1 A SOLUCAO DO PVI

    write.full.string (screen , "componente.")
    write.int (screen , 1 , 1)
    write.full.string (screen , "=")
    write.real64 (screen , y[1] , 2 , 10)
    newline(screen)
    newline(screen)
:
}}

```


REFERÊNCIAS

[Alexander,77] : R. Alexander , Diagonally implicit Runge-Kutta methods for stiff ODEs , SIAM Journal of Numerical Analysis 14 , 1006-1021 . (1977)

[Butcher,87] : J.C. Butcher , The numerical analysis of ordinary differential equations , Runge-Kutta and general linear methods , John Wiley and Sons . (1987)

[Butcher,Cash,89] : J.C. Butcher e J.R. Cash , Some recent developments on numerical initial value problems : a survey , Applied Numerical Mathematics , 5 , 3-18 . (1989)

[Dekker,Verwer,84] : K. Dekker e J.G. Verwer , Stability of Runge-Kutta methods for stiff nonlinear differential equations , North Holland . (1984)

[Hairer,Lubich,Roche,88] : E. Hairer , Ch. Lubich e M. Roche , Error of Runge-Kutta methods for stiff problems studied via differential algebraic equations , BIT 28 , 678-700 . (1988)

[Hairer,Norsett,Wanner,86] : E. Hairer ; S.P. Norsett e G. Wanner , Solving ordinary differential equations I : Nonstiff problems , Heidelberg : Springer-Verlag . (1986)

[Houwen,Sommeijer,90] : P.J. van der Houwen e B.P. Sommeijer , Iterated Runge-Kutta methods on parallel computers , Report NM-R9001 , Center for Mathematics and Computer Science , Amsterdam , Holland . (1990)

[Houwen,Sommeijer,Couzy,89] : P.J. van der Houwen ; B.P. Sommeijer e W. Couzy , Embedded diagonally implicit Runge-Kutta algorithms on parallel computers , Report NM-R8912 , Center for Mathematics and Computer Science , Amsterdam , Holland . (1989)

[Iserles, Norsett, 90] : A. Iserles e S.P. Norsett , On the theory of parallel Runge-Kutta methods , IMA Journal of Numerical Analysis 10 , 463-488 . (1990)

[Jackson, Norsett, 90] : K.R. Jackson e S.P. Norsett , The potencial for parallelism in Runge-Kutta methods , Technical Report No.239/90 , Computer Science Department , University of Toronto , Canada , M5S 1A4 . (1990)

[Lambert, 73] : J.D. Lambert , Computatinal methods in ordinary differential equations , John Wiley and Sons . (1973)

[Norsett, 74] : S.P. Norsett , Semi-explicit Runge-Kutta methods , Ph.D. Dissertation , University of Dundee . (1974)

[Poutain, May, 87] : D. Poutain e D. May , A tutorial introduction to OCCAM programming , BSP - Professional Books . (1987)

[Wolfbrandt, 77] : A. Wolfbrandt , A study of Rosenbrock processes with respect to order conditions and stiff stability , Ph.D. Thesis , Chalmers University of Technology , Goteborg .