UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA APLICADA

Polinômios ortogonais e a integração numérica na esfera unitária

por

Josadaque da Silva Nenê

Dissertação submetida como requisito parcial
para a obtenção do grau de

Mestre em Matemática Aplicada

Prof[a]. Dr[a]. Liliane Basso Barichello
Orientadora

Prof. Dr. Rudnei Dias da Cunha
Coorientador

Porto Alegre, 18 de fevereiro de 2020.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Nenê, Josadaque da Silva

Polinômios ortogonais e a integração numérica na esfera unitária / Josadaque da Silva Nenê.—Porto Alegre: PPGMAp da UFRGS, 2020.

251 p.: il.

Dissertação (Mestrado) —Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Matemática Aplicada e Estatística, Porto Alegre, 2020.
Orientadora: Barichello, Liliane Basso; Coorientador: da Cunha, Rudnei Dias

Dissertação: Matemática Aplicada,
Polinômios Ortogonais, Quadratura Gaussiana, Equação de Transporte, Quadratura QR, Integração Numérica

Polinômios ortogonais e a integração numérica na esfera unitária

por

Josadaque da Silva Nenê

Dissertação submetida ao Programa de Pós-Graduação em Matemática Aplicada do Instituto de Matemática e Estatística da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de

Mestre em Matemática Aplicada

Linha de Pesquisa: Teoria de transporte de partículas

Orientadora: Profª. Drª. Liliane Basso Barichello

Coorientador: Prof. Dr. Rudnei Dias da Cunha

Banca examinadora:

Profª. Drª. Cleonice Fatima Bracciali
IBILCE/UNESP

Profª. Drª. Patrícia Rodrigues
UFSM

Profª. Drª. Eliete Biasotto Hauser
PUCRS

Prof. Dr. Rudnei Dias da Cunha
DMPA/UFRGS

Dissertação apresentada e aprovada em
18 de fevereiro de 2020.

Prof. Dr. Esequia Sauter
Coordenador

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

$P_N T_N$	Quadratura Legendre-Chebyshev Quadrangular
$P_N T_N S_N$	Quadratura Legendre-Chebyshev Triangular
QR	Quadratura *Quadruple Range*
LQ_N	Quadratura *Level Symmetric*
S_N	Solução em ordenadas discretas
$QRS45m\_Q_{N_\theta}$	Quadratura QR, *flag azimutal = 1* com acoplamento quadrangular, gerada através da rotina *fn_qrs_full_tanh_sinh*
$QRA45m\_Q_{N_\theta}$	Quadratura QR, *flag azimutal = 2* com acoplamento quadrangular, gerada através da rotina *fn_qrs_full_tanh_sinh*
$QRJ45m\_Q_{N_\theta}$	Quadratura QR, *flag azimutal = 5* com acoplamento quadrangular, gerada através da rotina *fn_qrs_full_tanh_sinh*
$QRS90m\_Q_{N_\theta}$	Quadratura QR, *flag azimutal = 3* com acoplamento quadrangular, gerada através da rotina *fn_qrs_full_tanh_sinh*
$QRJ90m\_Q_{N_\theta}$	Quadratura QR, *flag azimutal = 6* com acoplamento quadrangular, gerada através da rotina *fn_qrs_full_tanh_sinh*
$QRS45m\_T_{N_\theta}$	Quadratura QR, *flag azimutal = 1* com acoplamento triangular, gerada através da rotina *fn_qrs_full_tanh_sinh*
$QRA45m\_T_{N_\theta}$	Quadratura QR, *flag azimutal = 2* com acoplamento triangular, gerada através da rotina *fn_qrs_full_tanh_sinh*

$QRJ45m\_T_{N_\theta}$	Quadratura QR, *flag azimutal = 5* com acoplamento triangular, gerada através da rotina *fn_qrs_full_tanh_sinh*
$QRS90m\_T_{N_\theta}$	Quadratura QR, *flag azimutal = 3* com acoplamento triangular, gerada através da rotina *fn_qrs_full_tanh_sinh*
$QRJ90m\_T_{N_\theta}$	Quadratura QR, *flag azimutal = 1* com acoplamento triangular, gerada através da rotina *fn_qrs_full_tanh_sinh*

LISTA DE VARIÁVEIS E SÍMBOLOS

$\vec{\Omega}$	Vetor unitário na direção angular de movimento da partícula
$\vec{r}$	Vetor posição espacial da partícula
σ_t	Seção de choque macroscópica total
σ_s	Seção de choque de espalhamento diferencial
$Q(\vec{r}, \vec{\Omega})$	Fonte externa de nêutrons
M_t	Número de total de direções da quadratura
$\Psi(\vec{r}, \vec{\Omega})$	Fluxo angular das partículas
θ	Ângulo polar, $0 \leq \theta \leq \pi$
ϕ	Ângulo azimutal, $0 \leq \phi \leq 2\pi$
Y_k^n	Função harmônico esférico de grau k e ordem n
Y_k^{n*}	Complexo conjugado de Y_k^n
P_l	Polinômio de Legendre de grau l
Γ	Função Gamma
B	Função Beta
x_k	k-ésimo nó de uma quadratura gaussiana
w_k	k-ésimo peso de uma quadratura gaussiana
$\omega(x)$	Função peso de quadratura
$\langle \cdot, \cdot \rangle$	Produto interno
π_l	Polinômio ortogonal de grau l
$E_n(f)$	Erro de truncamento

T_l	Polinômio de Chebyshev de 1º espécie de grau l
w_k^c	k-ésimo peso da quadratura de Gauss-Chebyshev
w_k^l	k-ésimo peso da quadratura de Gauss-Legendre
N	Ordem de quadratura para os esquemas $P_N T_N$ e $P_N T_N S_N$
W	Peso da quadratura produto
M	Número de direções discretas por octante da esfera unitária
N_θ	Ordem da quadratura unidimensional da variável polar (QR)
w_k^p	k-ésimo peso da quadratura unidimensional da variável polar (QR)
N_ϕ	Ordem da quadratura unidimensional da variável azimutal (QR)
w_k^a	k-ésimo peso da quadratura unidimensional da variável azimutal (QR)
$F(x)$	Descrição do sistema não linear; vetor coluna contendo as funções expressas como "strings"(no método de Newton)
x_0	Vetor coluna; estimativa inicial da solução (no método de Newton)
τ_r	Fator multiplicativo da norma L-2 de $F(x)$ avaliada na estimativa inicial (no método de Newton)
τ_a	Fator aditivo na tolerância ($\tau_a > \tau_r$) (no método de Newton)
$kmax$	Número máximo de iterações no método de Newton
tol_erf	Tolerância para a quadratura *função erro*

$tol\_tanh - sinh$	Tolerância para a quadratura *tanh-sinh*
ε_M	Épsilon de máquina, isto é, o menor número representável para o qual $1 + \varepsilon_M \neq 1$

RESUMO

Neste trabalho, são abordados os esquemas de quadraturas para integração na esfera unitária denominados *Legendre-Chebyshev Quadrangular (P_NT_N)*, *Legendre-Chebyshev Triangular ($P_NT_NS_N$)* e *Quadruple Range (QR)*. Tais esquemas são definidos como o produto de duas quadraturas gaussianas unidimensionais, uma associada a variável polar e outra associada a variável azimutal, que definem a direção das partículas na equação de transporte.

As quadraturas P_NT_N e $P_NT_NS_N$ são construídas a partir das quadraturas de Gauss-Legendre e Gauss-Chebyshev, e se diferenciam pela forma como são combinadas as ordens das quadraturas unidimensionais utilizadas. Já a quadratura QR foi desenvolvida para o tratamento de problemas bidimensionais de transporte de partículas, e sua construção envolve a resolução de sistemas não lineares mal-condicionados. Contudo, as dificuldades encontradas na resolução dos sistemas não lineares são contornadas pela utilização de polinômios ortogonais não clássicos, transformando a obtenção dos nós e pesos da quadratura QR em um problema de autovalores de matrizes tridiagonais simétricas.

Seguindo as ideias apresentadas na literatura para a geração das quadraturas QR via polinômios ortogonais não clássicos, são desenvolvidas duas variantes, resultando em dois novos conjuntos de quadratura para a variável azimutal. Além delas, também é proposta uma quadratura produto baseada na QR, que leva em conta a presença de uma singularidade azimutal.

Como parte fundamental deste trabalho, é implementado na linguagem Fortran 95 versões em precisão simples, dupla e quádrupla dos algoritmos usados na

construção das quadraturas, produzindo uma biblioteca de funções para a geração das mesmas. Com esta biblioteca, é possível obter tais quadraturas de forma rápida e com ordem arbitrária. Também é analisado o desempenho dos diferentes esquemas de quadraturas ao integrar polinômios nos cossenos diretores, tanto no octante principal quanto na esfera unitária, observando assim que os códigos implementados geram resultados de precisão elevada.

Para a integração no octante principal, os esquemas QR que utilizam a quadratura *Azimutal-QRS*45 e *Azimutal-QRA*45 se mostraram os mais adequados considerando a precisão nos resultados obtidos e o baixo tempo computacional. Para a esfera unitária, os esquemas $P_N T_N$ e $QRJ45m\_Q_{N_\theta}$ apresentaram resultados mais precisos, juntamente de um baixo tempo computacional. Além disso, tanto no octante principal quanto na esfera unitária, os esquemas QR baseados na quadratura *Azimutal-QRS*45 mostraram resutados mais precisos e tempos computacionais semelhentes aos esquemas baseados na quadratura *Azimutal-QRA*45.

ABSTRACT

In this work, the quadrature schemes for integration in the unitary sphere called *Legendre-Chebyshev Quadrangular (P_NT_N)*, *Legendre-Chebyshev Triangular ($P_NT_NS_N$)* and *Quadruple Range (QR)* are addressed. Such schemes are defined as the product of two one-dimensional Gaussian quadratures, one associated with the polar variable and the other associated with the azimuth variable, which define the direction of the particles in the transport equation.

The P_nT_n and $P_nT_nS_n$ quadratures are obtained from the Gauss-Legendre and Gauss-Chebyshev quadratures but differ from these in how are combined the degrees of the unidimensional quadrature used. On the other hand, the QR quadrature has been developed to deal specifically with bidimensional particle transport problems but its obtaining requires the solution of ill-conditioned nonlinear systems of equations. This difficulty is overcome by using non-classical orthogonal polynomials that have allowed us to obtain the roots and weights of the QR quadrature from the solution of a well-conditioned, tridiagonal symmetric eigenvalue problem.

Following the ideas presented in the generation of QR quadratures via non-classical orthogonal polynomials, two variants are developed, resulting in two new sets of quadrature for the azimuth variable. In addition to them, a quadrature product based on the QR quadrature is also proposed, which considers the presence of an azimuth singularity.

As a fundamental part of this work, a library of functions written in Fortran 95 provides implementations of the algorithms that describe the quadra-

ture schemes. These functions generate, in single-, double- and quadruple-precision, the quadratures nodes and weights, making it possible to obtain such quadratures quickly and in an arbitrary order. Using our implementations, the performance of the different quadrature schemes is also analyzed when integrating polynomials in the cosine directors, both in the principal octant and in the unitary sphere, thus observing that the implemented codes generate high precision results.

For integration in the principal octant, the QR schemes that use the *Azimutal-QRS*45 and *Azimutal-QRA*45 quadrature proved to be the most adequate, presenting better numerical results than the others. For the unit sphere, the $P_N T_N$ and $QRJ45m\_Q_{N_\theta}$ schemes presented the best results, followed by the $QRS45m\_Q_{N_\theta}$ quadrature. Also, we note that both in the principal octant and in the unit sphere, the QR schemes based on the *Azimutal-QRS*45 quadrature proved to be superior to the schemes based on the *Azimutal-QRA*45 quadrature.

1 INTRODUÇÃO

Durante seus estudos na teoria cinética dos gases, o físico Ludwig Boltzmann formulou o que passou a ser conhecida em diversas áreas da ciência como Equação de Boltzmann [12]. Tal equação é utilizada para modelar matematicamente vários fenômenos físicos, e suas aplicações se estendem a reatores nucleares [11, 26, 49], transporte de radiação [17, 33, 39], dinâmica de gases rarefeitos [32, 48, 67, 68], análises tomográficas [44, 47], entre outras.

A equação de transporte, versão linear da Equação de Boltzmann, na forma integro-diferencial representa um balanço matemático entre a produção e perda de partículas neutras, descrevendo quantitativamente a distribuição espacial, direcional, energética e temporal das partículas em meios materiais [31]. Tal equação, em sua forma geral, é dependente de sete variáveis independentes: três espaciais, duas angulares, uma energética e uma temporal. O tratamento analítico desta equação é muito complexo e métodos como o proposto por Case, método das soluções elementares [16, 31], podem ser utilizados apenas em casos idealizados. Devido a isso, métodos numéricos com enfoque probabilístico e determinístico têm sido usados na obtenção de sua solução. No primeiro caso, métodos como o de Monte Carlo [41], têm como objetivo resolver aproximadamente o problema exato incluindo vários aspectos físicos na modelagem. Já nos métodos determinísticos, a equação de transporte é tratada de forma aproximada e soluções precisas são buscadas. Formas aproximadas da equação de transporte íntegro-diferencial se fundamentam na discretização das variáveis que compõem o espaço de fase. A variável energética, no caso de aplicações nucleares, por exemplo, geralmente é discretizada através da aproximação multi-grupo [26], en-

quanto que as variáveis que compõem o domínio espacial são discretizadas através de diferentes métodos, cada qual com sua abordagem individual em vários sistemas de coordenadas, dos quais citamos: o método de diferenças finitas [42], o método dos volumes finitos [56], o método dos elementos finitos [58], e os métodos nodais [5, 55]. Por sua vez, as variáveis que compõem o domínio angular são frequentemente tratadas através do método dos harmônicos esféricos [28, 29, 57] ou do método das ordenadas discretas [8, 10, 17, 31].

Devido a sua versatilidade e precisão na solução de problemas da teoria de transporte, destacamos o método das ordenadas discretas. Este método foi proposto por Wick [74] na resolução de problemas de transporte de nêutrons e, posteriormente, foi desenvolvido e utilizado por Chandrasekhar [17] em seus estudos de transferência radiativa. Tem como base a discretização da direção das partículas e, consequentemente, aproximação do termo integral da equação de transporte íntegro-diferencial por uma quadratura numérica, reduzindo a equação de transporte a um sistema de equações diferenciais. Ele também é frequentemente utilizado nos principais códigos computacionais em teoria de transporte de nêutrons [53, 66, 69].

Contudo, utilizando o método das ordenadas discretas no tratamento das variáveis angulares, surge o chamado efeito raio [57], que está associado a discretização das variáveis angulares, e aparece independentemente do esquema de quadratura escolhido para discretizar o termo integral da equação de transporte. Muitas alternativas para reduzir ou eliminar o efeito raio na solução das equações em ordenadas discretas têm sido propostas e estudadas [24, 52, 57, 62, 63]. Neste contexto, o estudo de esquemas de quadraturas numéricas na esfera unitária é um tópico de grande interesse, visto que, o esquema de quadratura clássico utilizado na área, a quadratura *Level Symmetric* (LQ_N) [57], apresenta uma limitação quanto a ordem

utilizada, de modo que, para ordens $N > 20$ (o que representa 55 direções discretas por octante da esfera unitária), pesos negativos começam a surgir, implicando em soluções fisicamente impossíveis [54]. Este tópico tem sido de interesse do grupo de pesquisa [9, 20, 64, 73]. Tais estudos indicaram a necessidade da geração de códigos que possam fornecer dados precisos a respeito destas quadraturas.

O objetivo deste trabalho é dar continuidade ao estudo de aspectos teóricos e a implementação de códigos computacionais com precisão simples, dupla e quádrupla em linguagem Fortran 95 para geração de esquemas de quadraturas numéricas para a esfera unitária. Em particular, serão abordados os seguintes esquemas de quadraturas: Legendre-Chebyshev Quadrangular (P_NT_N), Legendre-Chebyshev Triangular ($P_NT_NS_N$) [15, 73] e as quadraturas *Quadruple Range (QR)* [1, 2, 70]. Tais esquemas de quadratura são definidos como o produto de duas quadraturas gaussianas unidimensionais, uma associada a variável polar e outra associada a variável azimutal, de modo que todos os pesos de quadratura são positivos, e com isso, eliminamos a restrição de ordem de quadratura apresentada pelo esquema clássico utilizado na área. As quadraturas P_NT_N e $P_NT_NS_N$ são construídas a partir das quadraturas unidimensionais de Gauss-Legendre e Gauss-Chebyshev, e se diferenciam pela forma como são combinados os graus das quadraturas unidimensionais utilizadas, bem como a escolha dos pesos. Com isso, a quadratura P_NT_N assume um padrão quadrangular de disposição das direções discretas na esfera unitária, enquanto que a quadratura $P_NT_NS_N$ assume um padrão triangular. Já a quadratura QR foi desenvolvida para o tratamento de problemas bidimensionais de transporte de nêutrons, e sua construção envolve a resolução de sistemas não lineares [1, 2]. Tais sistemas, principalmente o resultante da variável polar, são mal-condicionados, e devido a isso, somente esquemas de baixa ordem foram gerados numericamente

[1, 2]. Tais resultados foram, por exemplo, utilizados em [9, 73]. Para contornar o problema do mal condicionamento dos sistemas não lineares, Spence [70] propôs uma abordagem na qual determinamos os nós e pesos destas quadraturas unidimensionais a partir de polinômios ortogonais não clássicos. Dessa forma, a obtenção dos nós e pesos de quadratura é transformada em um problema de autovalores de uma matriz tridiagonal simétrica [70].

Este trabalho está estruturado de maneira que, no capítulo 2, apresentamos o teorema de Madsen [61], que estabelece uma relação entre a aproximação por quadratura do termo integral da equação de transporte e a convergência da solução em ordenadas discretas. Deduzimos também estimativas para o erro na aproximação do termo integral da equação de transporte de nêutrons por um esquema de quadratura numérica. No capítulo 3, estudamos as quadraturas gaussianas unidimensionais, apresentando sua relação com os polinômios ortogonais, o cálculo dos nós via problema de autovalores de uma matriz tridiagonal simétrica, formas de obter os pesos de quadratura e estimativas para o erro de truncamento destas quadraturas. Em particular, discutimos as quadraturas de Gauss-Legendre e Gauss-Chebyshev.

No capítulo 4, utilizamos a teoria das quadraturas gaussianas unidimensionais estudadas no capítulo anterior para construir os esquemas de quadratura para a esfera unitária, discutindo aspectos teóricos e computacionais, juntamente com estimativas para o erro de truncamento para os esquemas de quadratura multidimensionais. Dentre estas estimativas, apresentamos resultados que expressam, para o pior caso, a ordem de convergência do erro de truncamento em função do número de direções discretas usadas na aproximação. Já no capítulo 5, apresentamos três problemas testes e analisamos o desempenho numérico das quadraturas estudas. Em nossa análise, integramos monômios nos cossenos diretores para o domínio an-

gular correspondendo tanto ao octante principal da esfera unitária quanto a toda a esfera, cujo valor analítico foi obtido seguindo as referências [1, 73] e apresentado no capítulo 2.

Por fim, no capítulo 6, apresentamos as conclusões obtidas durante este trabalho, junto com um apêndice descrevendo todos os códigos computacionais implementados em Fortran 95, e tabelas apresentando os erros relativos mínimos e máximos para algumas ordens de quadratura discutidas no capítulo 5.

2 APROXIMAÇÃO POR QUADRATURAS DO TERMO INTEGRAL DA EQUAÇÃO DE TRANSPORTE

Neste capítulo, introduzimos a equação de transporte de nêutrons independente do tempo, que considera a distribuição das partículas em meio não multiplicativo, a um grupo de energia, de acordo com Duderstadt e Hamilton [26],

$$\vec{\Omega} \cdot \nabla \Psi(\vec{r}, \vec{\Omega}) + \sigma_t \Psi(\vec{r}, \vec{\Omega}) = \int_S \sigma_s(\vec{r}, \vec{\Omega'} \cdot \vec{\Omega}) \Psi(\vec{r}, \vec{\Omega'}) d\vec{\Omega'} + Q(\vec{r}, \vec{\Omega}), \qquad (2.1)$$

em que σ_t representa a seção de choque macroscópica total, $\sigma_s(\vec{r}, \vec{\Omega'} \cdot \vec{\Omega})$ a seção de choque macroscópica diferencial de espalhamento, $Q(\vec{r}, \vec{\Omega})$ é o termo de fonte externa de nêutros, e o termo integral da equação (2.1) é avaliado sobre todas as direções $\vec{\Omega} = (\Omega_x, \Omega_y, \Omega_z)$ da esfera unitária S, em que $\Psi(\vec{r}, \vec{\Omega})$ é o fluxo angular das partículas na posição $\vec{r} = (x, y, z)$ com

$$\vec{\Omega} \cdot \nabla \Psi(\vec{r}, \vec{\Omega}) = \Omega_x \frac{\partial \Psi}{\partial x} + \Omega_y \frac{\partial \Psi}{\partial y} + \Omega_z \frac{\partial \Psi}{\partial z} \ . \qquad (2.2)$$

A direção angular de movimento das partículas $\vec{\Omega} = (\Omega_x, \Omega_y, \Omega_z)$ é convenientemente descrita utilizando os cossenos diretores, como pode ser visto na figura 2.1, de modo que

$$\Omega_x = \mu = \cos\phi \sin\theta, \qquad (2.3)$$

$$\Omega_y = \eta = \sin\phi \sin\theta, \qquad (2.4)$$

e

$$\Omega_z = \xi = \cos\theta, \qquad (2.5)$$

sendo que os mesmos satisfazem

$$\Omega_x^2 + \Omega_y^2 + \Omega_z^2 = 1. \tag{2.6}$$

Neste trabalho, θ é o ângulo polar medido a partir do eixo z e ϕ é o ângulo azimutal medido a partir do eixo x.

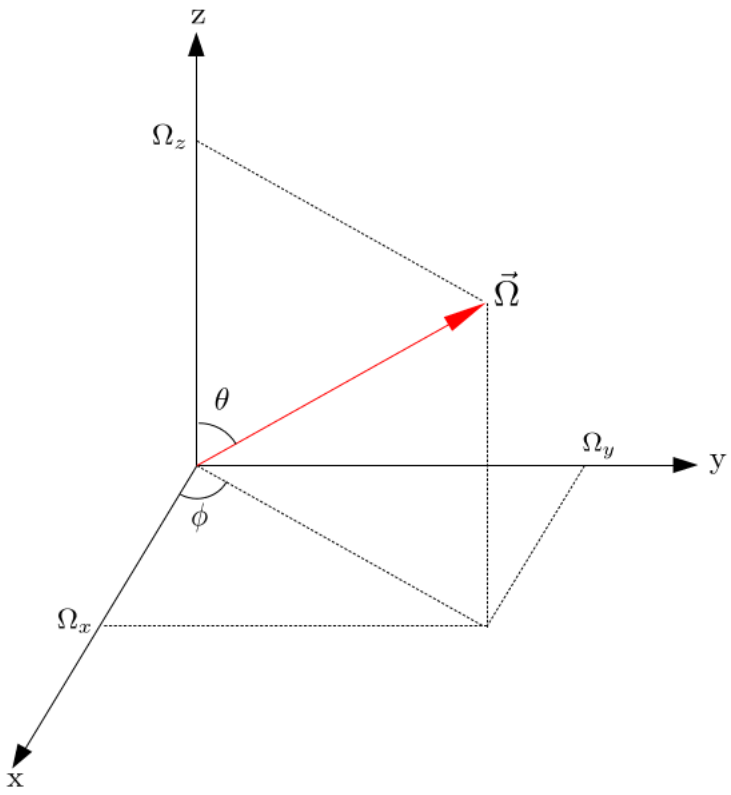


Figura 2.1 Sistema de Coordenadas Angulares

Como mencionado anteriormente, no método das ordenadas discretas as direções angulares são discretizadas e o termo integral da equação de transporte é aproximado por uma quadratura numérica. Neste contexto, um importante resultado é o teorema de Madsen [61], que é discutido na seção 2.1. Na seção 2.2, fazemos

uma estimativa do erro na aproximação do termo integral da equação de transporte bidimensional por um esquema de quadratura numérica nas variáveis angulares.

2.1 Teorema de Madsen e Solução em Ordenadas Discretas

Madsen [61] demonstrou que a solução em ordenadas discretas (S_N) da equação de transporte de nêutrons converge pontualmente para a solução exata da equação de transporte, desde que [61]:

1. o número médio de partículas secundárias por colisão seja menor que 1;
2. o erro de truncamento da quadratura tenda a zero quando o número de direções discretas tende a infinito;

e também estabelece uma relação entre o erro na solução S_N e o erro de truncamento da quadratura [43, 61].

O erro na solução S_N é limitado pelo erro de truncamento da quadratura, desde que [61]:

1. $\sigma_t(\vec{r})$, $\sigma_s(\vec{r}, \vec{\Omega}_m \cdot \vec{\Omega}_n)$ e $Q(\vec{r}, \vec{\Omega}_m)$ são contínuas por partes no espaço;
2. a seção de choque de absorção local satisfaz:

$$\sigma_{a,m}(\vec{r}) = \sigma(\vec{r}) - \sum_{n=1}^{M_t} w_m \sigma_s(\vec{r}, \vec{\Omega}_m \cdot \vec{\Omega}_n) \geq \Sigma_0 > 0, \qquad (2.7)$$

3. $\sigma_s(\vec{r}, \vec{\Omega}_m \cdot \vec{\Omega}_n) \geq 0$;

para todo $\vec{r} \in \mathbb{D}$, $m, n = 1, 2, \ldots, M_t$, e M_t é número total de direções discretas da quadratura. Em sua prova, Madsen considerou $\mathbb{D}$ sendo o domínio cartesiano tridimensional e também que a equação e a solução de transporte são espacialmente contínuas, e relacionou o erro da solução S_N com o erro da quadratura por [61]

$$||\epsilon||_\infty \leq \frac{1}{\Sigma_0}||\chi||_\infty, \tag{2.8}$$

em que $||\cdot||_\infty$ denota a norma L_∞, ϵ é o vetor representando o erro na solução aproximada,

$$\epsilon_m(\vec{r}) = \Psi(\vec{r}, \vec{\Omega}_m) - \Psi_m(\vec{r}), \tag{2.9}$$

e χ é o vetor de erro de truncamento da quadratura,

$$\chi_m(\vec{r}) = \int_S \sigma(\vec{r}, \vec{\Omega}_m \cdot \vec{\Omega}')\Psi(\vec{r}, \vec{\Omega}')d\vec{\Omega}' - \sum_{n=1}^{M_t} w_m \sigma_s(\vec{r}, \vec{\Omega}_m \cdot \vec{\Omega}_n)\Psi(\vec{r}, \vec{\Omega}_n), \tag{2.10}$$

com $m = 1, 2, \ldots, M_t$, $\Psi(\vec{r}, \vec{\Omega}_m)$ a solução exata da equação de transporte e $\Psi_m(\vec{r})$ a aproximação S_N.

Dessa forma, a estimativa dada pela equação (2.10) é extremamente relevante para análise do erro nas soluções da equação de transporte obtidas pelo método das ordenadas discretas. Neste sentido, apresentamos uma análise preliminar desta estimativa a seguir.

2.2 Estimativa de Erro na Aproximação por Quadratura do Termo Integral da Equação de Transporte

Visando a análise do erro na aproximação do termo integral da equação de transporte por meio do uso de quadraturas numéricas, inicialmente, expandimos

o fluxo angular presente no termo integral da equação (2.1) em termos das funções harmônicos esféricos como [73]

$$\Psi(\vec{r},\vec{\Omega'}) = \sum_{k=0}^{\infty}\sum_{n=-k}^{k} \Upsilon_k^n(\vec{r}) Y_k^n(\vec{\Omega'}), \tag{2.11}$$

em que

$$\Upsilon_k^n(\vec{r}) = \int_S \Psi(\vec{r},\vec{\Omega'}) Y_k^{n*}(\vec{\Omega'}) d\vec{\Omega'}. \tag{2.12}$$

Aqui, Y_k^n é chamada de função harmônico esférico de grau k e ordem n, e Y_k^{n*} denota o complexo conjugado de Y_k^n.

Substituindo a equação (2.11) na integral da equação (2.1), obtemos

$$\int_S \sigma_s(\vec{r},\vec{\Omega'}\cdot\vec{\Omega})\Psi(\vec{r},\vec{\Omega'})d\vec{\Omega'} = \sum_{k=0}^{\infty}\sum_{n=-k}^{k} \Upsilon_k^n(\vec{r}) \int_S \sigma_s(\vec{r},\vec{\Omega'}\cdot\vec{\Omega}) Y_k^n(\vec{\Omega'}) d\vec{\Omega'}. \tag{2.13}$$

Agora, representamos a seção de choque diferencial, $\sigma_s(\vec{r},\vec{\Omega'}\cdot\vec{\Omega})$, da equação (2.13) em termos dos polinômios de Legendre em [57]

$$\sigma_s(\vec{r},\vec{\Omega'}\cdot\vec{\Omega}) = \sum_{l=0}^{\infty}(2l+1)\sigma_{sl}(\vec{r}) P_l(\vec{\Omega'}\cdot\vec{\Omega}). \tag{2.14}$$

Como

$$\vec{\Omega'} = \sin\theta'\cos\phi'\,\vec{i} + \sin\theta'\sin\phi'\,\vec{j} + \cos\theta'\,\vec{k} \tag{2.15}$$

e

$$\vec{\Omega} = \sin\theta\cos\phi\,\vec{i} + \sin\theta\sin\phi\,\vec{j} + \cos\theta\,\vec{k}\,, \tag{2.16}$$

temos que

$$\vec{\Omega'}\cdot\vec{\Omega} = \sin\theta'\cos\phi'\sin\theta\cos\phi + \sin\theta'\sin\phi'\sin\theta\sin\phi + \cos\theta'\cos\theta$$

$$= \cos\theta' \cos\theta + \sin\theta' \sin\theta(\cos\phi' \cos\phi + \sin\phi' \sin\phi)$$
$$= \cos\theta' \cos\theta + \sin\theta' \sin\theta \cos(\phi' - \phi). \tag{2.17}$$

Usando a equação (2.17) e o teorema da adição de harmônicos esféricos, expressamos o polinômio de Legendre de grau l em $\vec{\Omega'} \cdot \vec{\Omega}$ através do produto de dois harmônicos esféricos com coordenadas angulares (θ, ϕ) e (θ', ϕ') como [57]

$$P_l(\vec{\Omega'} \cdot \vec{\Omega}) = \frac{1}{2l+1} \sum_{m=-l}^{l} Y_l^{m*}(\vec{\Omega'}) Y_l^m(\vec{\Omega}), \tag{2.18}$$

em que

$$Y_l^m(\vec{\Omega}) = \sqrt{\frac{(2l+1)(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi} \tag{2.19}$$

e

$$Y_l^{m*}(\vec{\Omega'}) = \sqrt{\frac{(2l+1)(l-m)!}{(l+m)!}} P_l^m(\cos\theta') e^{-im\phi'}. \tag{2.20}$$

Por fim, podemos escrever o termo da seção de choque de espalhamento diferencial $\sigma_s(\vec{r}, \vec{\Omega'} \cdot \vec{\Omega})$ como

$$\sigma_s(\vec{r}, \vec{\Omega'} \cdot \vec{\Omega}) = \sum_{l=0}^{\infty} \sigma_{sl}(\vec{r}) \sum_{m=-l}^{l} Y_l^{m*}(\vec{\Omega'}) Y_l^m(\vec{\Omega}) \tag{2.21}$$

e a integral dada pela equação (2.13) torna-se

$$\int_S \sigma_s(\vec{r}, \vec{\Omega'} \cdot \vec{\Omega}) \Psi(\vec{r}, \vec{\Omega'}) d\vec{\Omega'} =$$
$$\sum_{l=0}^{\infty} \sigma_{sl}(\vec{r}) \sum_{m=-l}^{l} Y_l^m(\vec{\Omega}) \sum_{k=0}^{\infty} \sum_{n=-k}^{k} \Upsilon_k^n(\vec{r}) \int_S Y_l^{m*}(\vec{\Omega'}) Y_k^n(\vec{\Omega'}) d\vec{\Omega'}. \tag{2.22}$$

Ainda utilizamos a propriedade de ortogonalidade dos harmônicos esféricos [57]

$$\int_S Y_l^{m*}(\vec{\Omega'}) Y_k^n(\vec{\Omega'}) d\vec{\Omega'} = \delta_{lk} \delta_{mn} \tag{2.23}$$

com

$$\delta_{ij} = \begin{cases} 1, \text{ se } i = j, \\ 0, \text{ caso contrário,} \end{cases} \tag{2.24}$$

a equação (2.22) pode ser escrita como

$$\int_S \sigma_s(\vec{r}, \vec{\Omega'} \cdot \vec{\Omega}) \Psi(\vec{r}, \vec{\Omega'}) d\vec{\Omega'} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sigma_{sl}(\vec{r}) Y_l^m(\vec{\Omega}) \Upsilon_l^m(\vec{r}). \tag{2.25}$$

Finalmente, levando em conta a equação (2.12) temos

$$\int_S \sigma_s(\vec{r}, \vec{\Omega'} \cdot \vec{\Omega}) \Psi(\vec{r}, \vec{\Omega'}) d\vec{\Omega'} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sigma_{sl}(\vec{r}) Y_l^m(\vec{\Omega}) \int_S \Psi(\vec{r}, \vec{\Omega'}) Y_l^{m*}(\vec{\Omega'}) d\vec{\Omega'}. \tag{2.26}$$

Em problemas de transporte bidimensionais, o último termo da equação (2.2) é desprezado, uma vez que o fluxo angular e todos os momentos do fluxo não variam com z. Com isso, a partir das expressões obtidas nas equações (2.2) e (2.26), escrevemos a equação de transporte (2.1), para problemas bidimensionais em geometria cartesiana, como

$$\mu \frac{\partial}{\partial x} \Psi(x, y, \vec{\Omega}) + \eta \frac{\partial}{\partial y} \Psi(x, y, \vec{\Omega}) + \sigma_t \Psi(x, y, \vec{\Omega}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sigma_{sl}(x, y) Y_l^m(\vec{\Omega}) \int_S \Psi(x, y, \vec{\Omega'}) Y_l^{m*}(\vec{\Omega'}) d\vec{\Omega'} + Q(x, y, \vec{\Omega}). \tag{2.27}$$

Para fins de estudo da aproximação da integral angular da equação bidimensional de transporte de nêutrons, suprimimos a notação da dependência espacial $\vec{r} = (x, y)$ da equação (2.26). Dessa forma, o integrando da equação (2.26) passa a

ser escrito como uma função dependente somente de $\overrightarrow{\Omega'} = (\Omega'_x, \Omega'_y)$. Então, definimos

$$\Psi(\overrightarrow{\Omega'})Y_l^{m*}(\overrightarrow{\Omega'}) = f_{lm}(\Omega'_x, \Omega'_y) \tag{2.28}$$

e, substituindo na equação (2.26), obtemos

$$\int_S \sigma_s(\overrightarrow{r}, \overrightarrow{\Omega'} \cdot \overrightarrow{\Omega})\Psi(\overrightarrow{r}, \overrightarrow{\Omega'})d\overrightarrow{\Omega'} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \sigma_{sl}(\overrightarrow{r})Y_l^m(\overrightarrow{\Omega}) \int_S f_{lm}(\Omega'_x, \Omega'_y)d\overrightarrow{\Omega'}. \tag{2.29}$$

Buscamos uma quadratura numérica que seja capaz de aproximar o termo integral dado pela equação (2.29) com máxima precisão possível, ou seja,

$$\int_S g(\overrightarrow{\Omega'})d\overrightarrow{\Omega'} \approx \sum_{\tilde{m}=0}^{N} \omega_{\tilde{m}} g(\overrightarrow{\Omega'}_{\tilde{m}}), \tag{2.30}$$

e, consequentemente, os momentos angulares dados pela equação (2.12), e expressos pela equação (2.26), isto é,

$$\sum_{\tilde{m}=0}^{N} \omega_{\tilde{m}} g(\overrightarrow{\Omega'}_{\tilde{m}}) = \sum_{\tilde{m}=0}^{N} \omega_{\tilde{m}} \Psi(\overrightarrow{\Omega'}_{\tilde{m}})Y_l^{m*}(\overrightarrow{\Omega'}_{\tilde{m}}) \approx \int_S \Psi(\overrightarrow{\Omega'})Y_l^{m*}(\overrightarrow{\Omega'})d\overrightarrow{\Omega'} = \Upsilon_l^m. \tag{2.31}$$

Expandimos a função $f_{lm}(\Omega'_x, \Omega'_y)$ definida pela equação (2.28) em séries de Taylor de duas variáveis centradas no ponto $(\Omega'_x = 0, \Omega'_y = 0)$, e obtemos a seguinte expressão para a integral do lado direito da equação (2.29):

$$\begin{aligned}
\int_S f_{lm}(\Omega'_x, \Omega'_y)d\overrightarrow{\Omega'} &= \int_S \left[\sum_{n=0}^{\infty}\frac{1}{n!}\sum_{i=0}^{n}\frac{n!}{i!(n-i)!}(\Omega'_x)^i(\Omega'_y)^{n-i}\frac{\partial^n}{\partial\Omega_x'^i\partial\Omega_y'^{n-i}}f_{lm}(0,0)\right]d\overrightarrow{\Omega'} \\
&= f_{lm}(0,0)\int_S 1 d\overrightarrow{\Omega'} + \frac{\partial}{\partial\Omega'_x}f_{lm}(0,0)\int_S \Omega'_x d\overrightarrow{\Omega'} \\
&\quad \frac{\partial}{\partial\Omega'_y}f_{lm}(0,0)\int_S \Omega'_y d\overrightarrow{\Omega'} + \frac{1}{2}\frac{\partial^2}{\partial\Omega_x'^2}f_{lm}(0,0)\int_S (\Omega'_x)^2 d\overrightarrow{\Omega'} + \\
&\quad + \frac{\partial^2}{\partial\Omega'_x\partial\Omega'_y}f_{lm}(0,0)\int_S \Omega'_x\Omega'_y d\overrightarrow{\Omega'} + \frac{1}{2}\frac{\partial^2}{\partial\Omega_y'^2}f_{lm}(0,0)\int_S (\Omega'_y)^2 d\overrightarrow{\Omega'} + \cdots .
\end{aligned} \tag{2.32}$$

Agora, aproximamos a integral dada pela equação (2.32) através da quadratura numérica dada pela equação (2.31), obtendo assim

$$\sum_{\tilde{m}=0}^{N}\omega_{\tilde{m}}f_{lm}(\Omega'_{x\tilde{m}},\Omega'_{y\tilde{m}})=\sum_{\tilde{m}=0}^{N}\omega_{\tilde{m}}\left[\sum_{n=0}^{\infty}\frac{1}{n!}\sum_{i=0}^{n}\frac{n!}{i!(n-i)!}(\Omega'_{x\tilde{m}})^{i}(\Omega'_{y\tilde{m}})^{n-i}\frac{\partial^{n}}{\partial\Omega_{x}^{\prime i}\partial\Omega_{y}^{\prime n-i}}f_{lm}(0,0)\right]$$
$$=f_{lm}(0,0)\sum_{\tilde{m}=0}^{N}\omega_{\tilde{m}}+\frac{\partial}{\partial\Omega'_{x}}f_{lm}(0,0)\sum_{\tilde{m}=0}^{N}\omega_{\tilde{m}}\Omega'_{x\tilde{m}}+$$
$$+\frac{\partial}{\partial\Omega'_{y}}f_{lm}(0,0)\sum_{\tilde{m}=0}^{N}\omega_{\tilde{m}}\Omega'_{y\tilde{m}}+\frac{1}{2}\frac{\partial^{2}}{\partial\Omega_{x}^{\prime 2}}f_{lm}(0,0)\sum_{\tilde{m}=0}^{N}\omega_{\tilde{m}}(\Omega'_{x\tilde{m}})^{2}+$$
$$+\frac{\partial^{2}}{\partial\Omega'_{x}\partial\Omega'_{y}}f_{lm}(0,0)\sum_{\tilde{m}=0}^{N}\omega_{\tilde{m}}\Omega'_{x\tilde{m}}\Omega'_{y\tilde{m}}+\frac{1}{2}\frac{\partial^{2}}{\partial\Omega_{y}^{\prime 2}}f_{lm}(0,0)\sum_{\tilde{m}=0}^{N}\omega_{\tilde{m}}(\Omega'_{y\tilde{m}})^{2}+\cdots. \tag{2.33}$$

Note que, ao expandirmos a função $f_{lm}(\Omega'_x,\Omega'_y)$ em série de Taylor, foi possível reescrevê-la em termos de polinômios em Ω'_x e Ω'_y. Sendo assim, assumindo que a função $f_{lm}(\Omega'_x,\Omega'_y)$ possa ser escrita em termos de um polinômio de grau L em Ω'_x e Ω'_y, em que L é a soma das potências de Ω'_x e Ω'_y, procuramos uma quadratura que integre exatamente a integral polinomial para o maior valor de L possível. Portanto, se um conjunto de quadratura integrar exatamente tais funções polinomiais (monômios em Ω'_x e Ω'_y) até grau L, esperamos que esse conjunto forneça soluções precisas para o problema de interesse. Entendemos que, se o conjunto de quadratura não puder integrar exatamente essas funções, erros na solução da equação de transporte podem ser produzidos [73]. Dessa forma, de acordo com o que foi deduzido acima, é fundamental avaliarmos bem as integrais do tipo

$$\int_{S}(\Omega_x)^{l}(\Omega_y)^{m}d\vec{\Omega}=\int_{0}^{2\pi}\int_{0}^{\pi}(\sin\theta\cos\phi)^{l}(\sin\theta\sin\phi)^{m}\sin\theta d\theta d\phi,$$
$$l+m\leq L. \tag{2.34}$$

Seguindo as derivações de Tres [73] e Hu [43], podemos encontrar uma expressão analítica para a integral dada pela equação (2.34). Assim, temos que

$$\int_0^{2\pi}\int_0^{\pi}(\sin\theta\cos\phi)^l(\sin\theta\sin\phi)^m\sin\theta d\theta d\phi =$$
$$= \int_0^{2\pi}\int_0^{\pi}\sin^l\theta\cos^l\phi\sin^m\theta\sin^m\phi\sin\theta d\theta d\phi$$
$$= \int_0^{2\pi}\int_0^{\pi}(\sin^{l+m}\theta\sin\theta)(\cos^l\phi\sin^m\phi)d\theta d\phi$$
$$= \left\{\int_0^{2\pi}\cos^l\phi\sin^m\phi d\phi\right\}\times\left\{\int_0^{\pi}\sin^{l+m}\theta\sin\theta d\theta\right\}. \quad (2.35)$$

Note que, a integral dupla da equação (2.34) foi transformada no produto de duas integrais, sendo uma dependente do ângulo azimutal ϕ e a outra dependente do ângulo polar θ. Com isso, o problema de resolver analiticamente a integral dupla dada pela equação (2.34), é transformado no problema de resolver analiticamente duas integrais unidimensionais.

Tratamos primeiro da integral na variável azimutal. Para $\phi \in [0, \pi/2]$ temos que $\sin\phi \geq 0$ e $\cos\phi \geq 0$. Com isso, sejam $r\sin\phi = x$ e $r\cos\phi = y$, e $r \in [0, \infty)$. Logo,

$$\int_0^{\pi/2}\sin^m\phi\cos^l\phi d\phi = \frac{\int_0^{\pi/2}\sin^m\phi\cos^l\phi d\phi\int_0^{\infty}r^{m+l+1}e^{-r^2}dr}{\int_0^{\infty}r^{m+l+1}e^{-r^2}dr}$$
$$= \frac{\int_0^{\pi/2}\int_0^{\infty}\sin^m\phi\cos^l\phi r^{m+l+1}e^{-r^2}drd\phi}{\int_0^{\infty}r^{m+l+1}e^{-r^2}dr}. \quad (2.36)$$

Fazendo a mudança de variável $\delta = r^2$ na integral do denominador da equação (2.36), obtemos

$$\int_0^{\infty}r^{m+l+1}e^{-r^2}dr = \frac{1}{2}\int_0^{\infty}\delta^{(\frac{l+m}{2})}e^{-\delta}d\delta$$

$$= \frac{1}{2}\int_0^\infty \delta^{(\frac{l+m+2}{2}-1)} e^{-\delta} d\delta$$
$$= \frac{1}{2}\Gamma\left(\frac{l+m+2}{2}\right), \tag{2.37}$$

em que Γ é a função Gamma [25]. No numerador da equação (2.36), fazendo a mudança de variável $r\sin\phi = x$ e $r\cos\phi = y$, obtemos

$$\int_0^{\pi/2}\int_0^\infty \sin^m\phi\cos^l\phi r^{m+l+1} e^{-r^2} dr d\phi = \int_0^\infty\int_0^\infty x^m y^l e^{-(x^2+y^2)} dx dy$$
$$= \int_0^\infty x^m e^{-x^2} dx \int_0^\infty y^l e^{-y^2} dy. \tag{2.38}$$

Agora, fazendo as mudanças de variáveis $\alpha = x^2$ e $\beta = y^2$, as integrais da equação (2.38) tornam-se

$$\int_0^\infty x^m e^{-x^2} dx \int_0^\infty y^l e^{-y^2} dy = \left[\frac{1}{2}\int_0^\infty \alpha^{(\frac{m-1}{2}+1-1)} e^{-\alpha} d\alpha\right]\left[\frac{1}{2}\int_0^\infty \beta^{(\frac{l-1}{2}+1-1)} e^{-\beta} d\beta\right]$$
$$= \frac{1}{2}\Gamma\left(\frac{m+1}{2}\right)\frac{1}{2}\Gamma\left(\frac{l+1}{2}\right). \tag{2.39}$$

Com isso, obtemos

$$\int_0^{\pi/2} \sin^m\phi\cos^l\phi d\phi = \frac{\Gamma(\frac{m+1}{2})\Gamma(\frac{l+1}{2})}{2\Gamma(\frac{l+m+2}{2})}. \tag{2.40}$$

Considerando os quatro quadrantes da integral sobre $\phi \in [0, 2\pi]$ individualmente, temos

$$\int_{\pi/2}^{\pi} \sin^m\phi\cos^l\phi d\phi = (-1)^l \int_0^{\pi/2} \sin^m\phi\cos^l\phi d\phi \tag{2.41}$$
$$\int_{\pi}^{3\pi/2} \sin^m\phi\cos^l\phi d\phi = (-1)^l(-1)^m \int_0^{\pi/2} \sin^m\phi\cos^l\phi d\phi \tag{2.42}$$
$$\int_{3\pi/2}^{2\pi} \sin^m\phi\cos^l\phi d\phi = (-1)^m \int_0^{\pi/2} \sin^m\phi\cos^l\phi d\phi, \tag{2.43}$$

e concluímos que

$$\int_0^{2\pi} \sin^m \phi \cos^l \phi d\phi = [1+(-1)^l][1+(-1)^m]\frac{\Gamma(\frac{m+1}{2})\Gamma(\frac{l+1}{2})}{2\Gamma(\frac{l+m+2}{2})}. \tag{2.44}$$

Para o tratamento da integral na variável polar da equação (2.35), temos

$$\int_0^{\pi} \sin^{l+m}\theta \sin\theta d\theta = 2\int_0^{\pi/2} \sin^{l+m+2-1}\theta \cos^{1-1}\theta d\theta = B\left(\frac{l+m+2}{2}, \frac{1}{2}\right), \tag{2.45}$$

em que $B(x,y)$ é a função Beta [25]. Devido a relação entre as funções Beta e Gamma [25]

$$B\left(\frac{l+m+2}{2}, \frac{1}{2}\right) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}, \tag{2.46}$$

concluímos que

$$\int_0^{\pi} \sin^{l+m}\theta \sin\theta d\theta = \frac{\sqrt{\pi}\Gamma(\frac{l+m+2}{2})}{\Gamma(\frac{l+m+3}{2})}. \tag{2.47}$$

Combinando as equações (2.44) e (2.47), a expressão analítica para a integral definida na equação (2.35) é

$$\begin{aligned}\int_S \Omega_x^l \Omega_y^m d\vec{\Omega} &= \int_0^{2\pi}\int_0^{\pi} (\sin\theta\cos\phi)^l (\sin\theta\sin\phi)^m \sin\theta d\theta d\phi \\ &= [1+(-1)^l][1+(-1)^m]\sqrt{\pi}\frac{\Gamma(\frac{m+1}{2})\Gamma(\frac{l+1}{2})}{2\Gamma(\frac{l+m+3}{2})}.\end{aligned} \tag{2.48}$$

Através da equação (2.48), é possivel obter a solução exata para a integral polinomial em Ω_x e Ω_y de grau $L = l + m$, de forma que o erro de truncamento na aproximação via quadratura numérica pode ser estimado.

O próximo capítulo trará uma breve revisão sobre quadraturas gaussianas unidimensionais, que servirão de base para a obtenção de quadraturas produto para o caso bidimensional. O desempenho destas quadraturas produto será analisado com base no resultado acima.

3 QUADRATURAS GAUSSIANAS

O objetivo deste capítulo é trazer uma breve revisão sobre fórmulas de quadraturas gaussianas para o caso unidimensional. Definimos quadraturas gaussianas, de acordo com Kress [50], como:

Definição 3.1. *Uma fórmula de quadratura*

$$\int_a^b \omega(x) f(x) dx \approx \sum_{k=0}^{n} w_k f(x_k) \tag{3.1}$$

com $n+1$ nós de quadratura distintos é chamada ***fórmula de quadratura gaussiana*** *se ela integra exatamente todos os polinômios $q \in P_{2n+1}$, isto é:*

$$\int_a^b \omega(x) q(x) dx = \sum_{k=0}^{n} w_k q(x_k), \tag{3.2}$$

para todo polinômio q de grau $\leq 2n+1$.

Na definição (3.1), a função $\omega(x)$ é chamada de *função peso*, e satisfaz [75]:

1. $\omega(x)$ é uma definida em $[a, b]$;

2. $\omega(x) \geq 0$ para todo $x \in [a, b]$;

3. As integrais
$$\int_a^b \omega(x) x^n dx \equiv c_n \tag{3.3}$$
existem e são finitas para cada $n = 0, 1, 2, \ldots$, e $c_0 > 0$.

Tais fórmulas de quadratura desempenham um papel importante no campo da integração numérica, visto que as quadraturas gaussianas são as melhores fórmulas de quadratura, no sentido de, com n nós e pesos, ser capaz de integrar exatamente polinômios de grau $\leq 2n-1$ [23].

Usualmente, a obtenção dos nós e pesos da quadratura gaussiana exige a resolução de sistemas não lineares, mas isto pode ser evitado fazendo uso de polinômios ortogonais [4, 23, 30, 50, 71]. Sendo assim, a próxima seção deste capítulo será dedicada ao estudo dos polinômios ortogonais e, por fim, serão abordados dois casos específicos de quadraturas gaussianas: a quadratura de Gauss-Legendre e a quadratura de Gauss-Chebyshev.

3.1 Polinômios Ortogonais

Antes de iniciarmos a discussão sobre os polinômios ortogonais, são feitas as seguintes definições.

Definição 3.2. *Sejam p e q dois polinômios reais. Definimos o produto interno entre p e q, denotado por $\langle p, q\rangle$ como sendo*

$$\langle p, q\rangle = \int_a^b \omega(x)p(x)q(x)dx \ , \tag{3.4}$$

em que $\omega(x)$ denota uma função peso. A norma de q é definida como

$$||q|| = \left(\int_a^b \omega(x)q^2(x)dx\right)^{1/2} = \langle q, q\rangle^{1/2}. \tag{3.5}$$

A seguir, definimos o que são polinômios reais ortogonais e ortonormais.

Definição 3.3. *Os polinômios reais p e q são ditos polinômios ortogonais associados a função peso $\omega(x)$ e intervalo $[a, b]$, se*

$$\langle p, q\rangle = \int_a^b \omega(x)p(x)q(x)dx = 0. \tag{3.6}$$

Os polinômios reais p em um conjunto de polinômios são ortonormais se eles são mutuamente ortogonais e se

$$\langle p, p\rangle = \int_a^b \omega(x)p^2(x)dx = 1. \tag{3.7}$$

No decorrer do trabalho, sempre que nos refirirmos aos polinômios ortogonais (ou ortonormais), estaremos nos referindos a polinômios ortogonais (ou ortonormais) reais.

Uma forma de se obter polinômios ortogonais $\pi_0(x), \pi_1(x), \pi_2(x), \ldots$ é através da relação de recorrência que geram estes polinômios [30]. A relação de recorrência é dada no teorema a seguir.

Teorema 3.1. *Sejam $\pi_0(x), \pi_1(x), \pi_2(x), \ldots$ polinômios de graus $0, 1, 2, \ldots$, definidos por*

$$\begin{cases} \pi_{k+1}(x) & = (x - \alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x), \text{ para } k = 1, 2, \ldots \\ & \pi_0(x) = 1 \ , \ \pi_1(x) = x - \frac{\langle x,1\rangle}{\langle 1,1\rangle}, \end{cases} \tag{3.8}$$

em que

$$\alpha_k = \frac{\langle x\pi_k(x), \pi_k(x)\rangle}{\langle \pi_k(x), \pi_k(x)\rangle} \ , \ \beta_k = \frac{\langle \pi_k(x), \pi_k(x)\rangle}{\langle \pi_{k-1}(x), \pi_{k-1}(x)\rangle} \ , \text{ para } k = 1, 2, \ldots. \tag{3.9}$$

Os polinômios $\pi_0(x), \pi_1(x), \pi_2(x), \ldots$, assim definidos, são dois a dois ortogonais.

Demonstração: A demonstração é feita por indução, de acordo com Franco [30]. Para $\pi_0(x)$ e $\pi_1(x)$ é verdadeiro, pois

$$\begin{aligned}\langle \pi_0(x), \pi_1(x)\rangle &= \langle x - \frac{\langle x,1\rangle}{\langle 1,1\rangle}1, 1\rangle \\ &= \langle x,1\rangle - \frac{\langle x,1\rangle}{\langle 1,1\rangle}\langle 1,1\rangle \\ &= \langle x,1\rangle - \langle x,1\rangle \\ &= 0.\end{aligned}$$

Logo, $\pi_0(x)$ e $\pi_1(x)$ são ortogonais.

Suponhamos que $\langle \pi_i(x), \pi_j(x)\rangle = 0$ para $i \neq j$ e $i, j = 0, 1, 2, \ldots, k$. Provaremos que $\langle \pi_{k+1}(x), \pi_j(x)\rangle = 0$ para $j = 0, 1, 2, ..., k$. Dividiremos a prova em três partes.

a) Consideremos $j = k$. Temos

$$\begin{aligned}\langle \pi_{k+1}(x), \pi_k(x)\rangle &= \langle (x-\alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x), \pi_k(x)\rangle \\ &= \langle x\pi_k(x), \pi_k(x)\rangle - \alpha_k\langle \pi_k(x), \pi_k(x)\rangle - \beta_k\langle \pi_{k-1}(x), \pi_k(x)\rangle \\ &= \langle x\pi_k(x), \pi_k(x)\rangle - \frac{\langle x\pi_k(x), \pi_k(x)\rangle}{\langle \pi_k(x), \pi_k(x)\rangle}\langle \pi_k(x), \pi_k(x)\rangle \\ &= 0.\end{aligned}$$

Logo, $\pi_{k+1}(x)$ e $\pi_k(x)$ são ortogonais.

b) Para $j = k - 1$ temos

$$\begin{aligned}\langle \pi_{k+1}(x), \pi_{k-1}(x)\rangle &= \langle (x-\alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x), \pi_{k-1}(x)\rangle \\ &= \langle x\pi_k(x), \pi_{k-1}(x)\rangle - \alpha_k\langle \pi_k(x), \pi_{k-1}(x)\rangle - \beta_k\langle \pi_{k-1}(x), \pi_{k-1}(x)\rangle \\ &= \langle \pi_k(x), x\pi_{k-1}(x)\rangle - \beta_k\langle \pi_{k-1}(x), \pi_{k-1}(x)\rangle,\end{aligned}$$

desde que estamos utilizando o produto interno e a hipótese de indução. Note que,

$$\langle \pi_k(x), x\pi_{k-1}(x)\rangle = \langle \pi_k(x), \pi_k(x)\rangle.$$

Com efeito, utilizando a definição de $\pi_k(x)$, para $k = 1$, temos

$$\begin{aligned}
\langle \pi_1(x), x\pi_0(x)\rangle &= \langle \pi_1(x), x\rangle \\
&= \langle \pi_1(x), \pi_1(x) + \frac{\langle x\pi_0(x), \pi_0(x)\rangle}{\langle \pi_0(x), \pi_0(x)\rangle}\pi_0(x)\rangle \\
&= \langle \pi_1(x), \pi_1(x)\rangle + \frac{\langle x\pi_0(x), \pi_0(x)\rangle}{\langle \pi_0(x), \pi_0(x)\rangle}\langle \pi_1(x), \pi_0(x)\rangle \\
&= \langle \pi_1(x), \pi_1(x)\rangle,
\end{aligned}$$

desde que $\pi_0(x)$ e $\pi_1(x)$ são ortogonais. Para $k > 1$

$$\begin{aligned}
\langle \pi_k(x), x\pi_{k-1}(x)\rangle &= \langle \pi_k(x), \pi_k(x) + \alpha_{k-1}\pi_{k-1}(x) + \beta_{k-1}\pi_{k-2}(x)\rangle \\
&= \langle \pi_k(x), \pi_k(x)\rangle,
\end{aligned}$$

desde que, pela hipótese de indução, $\pi_k(x)$, $\pi_{k-1}(x)$ e $\pi_{k-2}(x)$ são ortogonais. Finalmente, temos que

$$\langle \pi_{k+1}(x), \pi_{k-1}(x)\rangle = \langle \pi_k(x), \pi_k(x)\rangle - \beta_k\langle \pi_{k-1}(x), \pi_{k-1}(x)\rangle = 0,$$

pois

$$\beta_k = \frac{\langle \pi_k(x), \pi_k(x)\rangle}{\langle \pi_{k-1}(x), \pi_{k-1}(x)\rangle}.$$

c) Vamos considerar $j = k-2, k-3, \ldots, 2, 1, 0$. Assim,

$$\begin{aligned}
\langle \pi_{k+1}(x), \pi_j(x)\rangle &= \langle (x-\alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x), \pi_j(x)\rangle \\
&= \langle \pi_k(x), x\pi_j(x)\rangle - \alpha_k\langle \pi_k(x), \pi_j(x)\rangle - \beta_k\langle \pi_{k-1}(x), \pi_j(x)\rangle \\
&= \langle \pi_k(x), x\pi_j(x)\rangle
\end{aligned}$$

$$= \langle \pi_k(x), \pi_{j+1}(x) + \alpha_j \pi_j(x) + \beta_j \pi_{j-1}(x) \rangle$$
$$= \langle \pi_k(x), \pi_{j+1}(x) \rangle + \alpha_j \langle \pi_k(x), \pi_j(x) \rangle + \beta_j \langle \pi_k(x), \pi_{j-1}(x) \rangle$$
$$= 0 \text{ , pois } j < k-1.$$

Logo, os polinômios são dois a dois ortogonais. ∎

Uma consequência importante do teorema (3.1) é que, a partir dele, podemos obter polinômios ortonormais [37]. De fato, suponha que temos um conjunto de polinômios π_k satisfazendo o teorema (3.1), então podemos obter polinômios ortonormais π_k^* pela normalização

$$\pi_k^*(x) = \frac{\pi_k(x)}{\langle \pi_k(x), \pi_k(x) \rangle^{1/2}}. \tag{3.10}$$

Substituindo a equação (3.10) na equação (3.8), temos

$$||\pi_{k+1}||\pi_{k+1}^*(x) = (x - \alpha_k)||\pi_k||\pi_k^*(x) - \beta_k||\pi_{k-1}||\pi_{k-1}^*(x) \tag{3.11}$$
$$\frac{||\pi_{k+1}||}{||\pi_k||}\pi_{k+1}^*(x) = (x - \alpha_k)\pi_k^*(x) - \beta_k \frac{||\pi_{k-1}||}{||\pi_k||}\pi_{k-1}^*(x). \tag{3.12}$$

Note que,

- $\beta_k \frac{||\pi_{k-1}||}{||\pi_k||} = \frac{||\pi_k||^2}{||\pi_{k-1}||^2} \frac{||\pi_{k-1}||}{||\pi_k||} = \frac{||\pi_k||}{||\pi_{k-1}||} = \sqrt{\beta_k}$;
- $\frac{||\pi_{k+1}||}{||\pi_k||} = \sqrt{\beta_{k+1}}$.

Assim, concluímos que

Teorema 3.2. *Sejam $\pi_0(x), \pi_1(x), \ldots$, polinômios ortogonais definidos conforme o teorema* (3.1). *Então, os polinômios $\pi_k^*(x) = \frac{\pi_k(x)}{||\pi_k||}$ são ortonormais, e satisfazem a*

seguinte relação de recorrência de três termos

$$\begin{cases} \sqrt{\beta_{k+1}}\pi^*_{k+1}(x) &= (x-\alpha_k)\pi^*_k(x) - \sqrt{\beta_k}\pi^*_{k-1}(x),, \text{ para } k = 1, 2, \ldots \\ & \pi^*_0(x) = \frac{1}{||\pi_0||} \ , \ \pi^*_1(x) = \frac{1}{\sqrt{\beta_1}}(x-\alpha_1)\pi^*_0(x) = \frac{\pi_1(x)}{||\pi_1||}, \end{cases} \tag{3.13}$$

em que α_k, β_k e π_k são definidos conforme o teorema (3.1).

Com isso, conhecendo os coeficientes α_k e β_k, $k = 1, 2, \ldots$, podemos determinar a sequência de polinômios ortonormais simplesmente tomando a raiz quadrada de β_k [37].

Além dos dois teoremas acima, os polinômios ortogonais gozam das seguintes propriedades:

Propriedade 3.1. *Sejam $\pi_0(x), \pi_1(x), \pi_2(x), \ldots$, polinômios ortogonais. Então, qualquer polinômio de grau menor ou igual a n pode ser escrito como combinação linear de $\pi_0(x), \pi_1(x), \pi_2(x), \ldots, \pi_n(x)$.*

Demonstração: Os polinômios $\pi_0(x), \pi_1(x), \pi_2(x), \ldots, \pi_n(x)$ constituem uma base para o espaço dos polinômios de grau menor ou igual a n. Assim, se $Q(x)$ é um polinômio da forma

$$Q(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n,$$

então $Q(x)$ pode ser escrito, através de uma mudança de base, como

$$Q(x) = b_0\pi_0(x) + b_1\pi_1(x) + \cdots + b_n\pi_n(x).$$

Com isso, está provada a propriedade. ∎

Propriedade 3.2. *Sejam $\pi_0(x), \pi_1(x), \pi_2(x), \ldots$, polinômios ortogonais. Então, $\pi_n(x)$ é ortogonal a qualquer polinômio de grau menor que n.*

Demonstração: Seja $Q(x)$ um polinômio de grau $n-1$. Pela propriedade anterior, temos que

$$Q(x) = b_0\pi_0(x) + b_1\pi_1(x) + \cdots + b_{n-1}\pi_{n-1}(x).$$

Consequentemente,

$$\begin{aligned}\langle Q(x), \pi_n(x)\rangle &= \langle b_0\pi_0(x) + b_1\pi_1(x) + \cdots + b_{n-1}\pi_{n-1}(x), \pi_n(x)\rangle \\ &= b_0\langle \pi_0(x), \pi_n(x)\rangle + \cdots + b_{n-1}\langle \pi_{n-1}(x), \pi_n(x)\rangle \\ &= 0,\end{aligned}$$

pois os polinômios $\pi_0(x), \pi_1(x), \pi_2(x), \ldots, \pi_n(x)$ são dois a dois ortogonais. ∎

Propriedade 3.3. *Sejam $\pi_0(x), \pi_1(x), \ldots$, polinômios ortogonais. Então $\pi_n(x)$ possui n raízes (reais) distintas em $[a, b]$.*

Demonstração: Dividiremos a demonstração em três partes:

1. $\pi_n(x)$ possui alguma raiz em $[a, b]$;
2. as raízes de π_n em $[a, b]$ são simples ;
3. as n raízes de $\pi_n(x)$ estão em $[a, b]$.

(1): Vamos supor, por absurdo, que π_n não possui raízes em $[a, b]$. Portanto em $[a, b]$, $\pi_n(x) \neq 0$. Logo,

$$\begin{aligned}\langle \pi_n(x), \pi_0(x)\rangle &= \int_a^b \omega(x)\pi_n(x)\pi_0(x)dx \\ &= \int_a^b \omega(x)\pi_n(x)dx \neq 0,\end{aligned}$$

desde que $\pi_0(x) = 1$, $\omega(x) \geq 0$ mas não identicamente nula, e $\pi_n(x) \neq 0$ em $[a, b]$. Mas $\pi_n(x)$ e $\pi_0(x)$ são ortogonais, e portanto $\langle \pi_n(x), \pi_0(x) \rangle = 0$. Portanto, é um absurdo supor que π_n não possui raízes em $[a, b]$.

(2): Vamos supor, por absurdo, que exista uma raiz de $\pi_n(x)$ que seja de multiplicidade 2. Seja x_1 essa raiz. Portanto,

$$\frac{\pi_n(x)}{(x - x_1)^2}$$

é um polinômio de grau $n - 2$. Assim, pela propriedade anterior,

$$\langle \pi_n(x), \frac{\pi_n(x)}{(x - x_1)^2} \rangle = 0.$$

Mas, usando as propriedades do produto interno, obtemos:

$$\begin{aligned} \langle \pi_n(x), \frac{\pi_n(x)}{(x - x_1)^2} \rangle &= \int_a^b \omega(x) \pi_n(x) \frac{\pi_n(x)}{(x - x_1)^2} dx \\ &= \int_a^b \omega(x) \frac{\pi_n(x)}{(x - x_1)} \frac{\pi_n(x)}{(x - x_1)} dx \\ &= \langle \frac{\pi_n(x)}{(x - x_1)}, \frac{\pi_n(x)}{(x - x_1)} \rangle \geq 0, \end{aligned}$$

com igualdade se e somente $\frac{\pi_n(x)}{(x - x_1)}$ for o polinômio nulo. Portanto, é um absurdo supor que os zeros de $\pi_n(x)$ em $[a, b]$ não são simples.

(3): Vamos supor, por absurdo, que existam somente j raízes de $\pi_n(x)$ em $[a, b]$, com $j < n$. Sejam $x_1, x_2, \ldots, x_j$ as raízes de $\pi_n(x)$ em $[a, b]$. Então, podemos escrever

$$\pi_n(x) = (x - x_1)(x - x_2) \ldots (x - x_j) q_{n-j}(x)$$

em que $q_{n-j} \neq 0$ em $[a, b]$. Logo, pela propriedade anterior, segue que

$$\langle \pi_n(x), (x - x_1)(x - x_2) \ldots (x - x_j) \rangle = 0.$$

Mas, usando as propriedades do produto interno, temos

$$\langle \pi_n(x), (x-x_1)(x-x_2)\dots(x-x_j)\rangle =$$
$$= \int_a^b \omega(x)(x-x_1)(x-x_2)\dots(x-x_j)q_{n-j}(x)(x-x_1)(x-x_2)\dots(x-x_j)dx$$
$$= \int_a^b \omega(x)(x-x_1)^2(x-x_2)^2\dots(x-x_j)^2 q_{n-j}(x)dx \neq 0.$$

Logo, é um absurdo supor que os n zeros de π_n não estão em $[a,b]$. ∎

Utilizando a relação de recorrência (3.1) que os polinômios ortogonais satisfazem, podemos provar o seguinte resultado, conhecido como fórmula de Christoffel-Darboux [37].

Teorema 3.3. *Sejam π_k^*, $k = 0, 1, \dots$ polinômios ortonormais. Então*

$$\sum_{i=0}^{k} \pi_i^*(x)\pi_i^*(y) = \sqrt{\beta_{k+1}}\frac{\pi_{k+1}^*(x)\pi_k^*(y) - \pi_k^*(x)\pi_{k+1}^*(y)}{x-y}, \text{ se } x \neq y, \tag{3.14}$$

e

$$\sum_{i=0}^{k} (\pi_i^*(x))^2 = \sqrt{\beta_{k+1}}[(\pi_{k+1}^*(x))'\pi_k^*(x) - (\pi_k^*(x))'\pi_{k+1}^*(x)]. \tag{3.15}$$

Demonstração: A demonstração é feita de acordo com Golub e Meurant [37]. Multiplicando a equação (3.13) por $\pi_k^*(y)$ obtemos

$$\sqrt{\beta_{k+1}}\pi_{k+1}^*(x)\pi_k^*(y) = (x-\alpha_k)\pi_k^*(x)\pi_k^*(y) - \sqrt{\beta_k}\pi_{k-1}^*(x)\pi_k^*(y). \tag{3.16}$$

Agora, multiplicamos novamente a equação (3.13) por $\pi_k^*(x)$ (trocando os papeis de x e y), obtendo assim

$$\sqrt{\beta_{k+1}}\pi_{k+1}^*(y)\pi_k^*(x) = (y-\alpha_k)\pi_k^*(y)\pi_k^*(x) - \sqrt{\beta_k}\pi_{k-1}^*(y)\pi_k^*(x). \tag{3.17}$$

Por fim, subtraindo as equações (3.16) e (3.17), obtemos

$$\begin{aligned}\sqrt{\beta_{k+1}}[\pi^*_{k+1}(x)\pi^*_k(y) - \pi^*_{k+1}(y)\pi^*_k(x)] &= \pi^*_k(y)\pi^*_k(x)(x - \alpha_k - y + \alpha_k) \\ &\quad + \sqrt{\beta_k}[\pi^*_{k-1}(y)\pi^*_k(x) - \pi^*_{k-1}(x)\pi^*_k(y)] \\ \sqrt{\beta_{k+1}}\frac{\pi^*_{k+1}(x)\pi^*_k(y) - \pi^*_{k+1}(y)\pi^*_k(x)}{x-y} &= \pi^*_k(y)\pi^*_k(x) + \sqrt{\beta_k}\frac{\pi^*_{k-1}(y)\pi^*_k(x) - \pi^*_{k-1}(x)\pi^*_k(y)}{x-y}.\end{aligned} \tag{3.18}$$

Note que a igualdade dada pela equação (3.18) se mantém para $k, k-1, k-2, \ldots, 0$. Logo, substituindo k por $0, 1, 2, \ldots, k$ e adicionando, concluímos que

$$\sum_{i=0}^{k} \pi^*_i(x)\pi^*_i(y) = \sqrt{\beta_{k+1}}\frac{\pi^*_{k+1}(x)\pi^*_k(y) - \pi^*_k(x)\pi^*_{k+1}(y)}{x-y}, \text{ se } x \neq y. \tag{3.19}$$

Para o caso $x = y$, o resultado é obtido fazendo $x \to y$. ∎

O próximo teorema, principal teorema desta seção, é o responsável por relacionar os polinômios ortogonais com as quadraturas gaussianas, definidas na definição (3.1) [30].

Teorema 3.4. *Sejam $\pi_0(x), \pi_1(x), \ldots$, polinômios ortogonais. Sejam $x_0, x_1, \ldots, x_n$ as raízes de $\pi_{n+1}(x)$. Se $f(x)$ é um polinômio de grau $\leq 2n+1$, então*

$$\int_a^b \omega(x)f(x)dx = \sum_{k=0}^{n} w_k f(x_k), \tag{3.20}$$

em que

$$w_k = \int_a^b \omega(x)l_k(x)dx \tag{3.21}$$

e

$$l_k(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)}. \tag{3.22}$$

Demonstração: Faremos a demonstração de acordo com Franco [30]. Como $x_0, x_1, \ldots, x_n$ são as raízes de $\pi_{n+1}(x)$, podemos escrever $\pi_{n+1}(x)$ como

$$\pi_{n+1}(x) = (x - x_0)(x - x_1)\ldots(x - x_n). \tag{3.23}$$

Seja $p_n(x)$ o polinômio de interpolação de $f(x)$ sobre $x_0, x_1, \ldots, x_n$ em $[a, b]$. Da teoria de interpolação polinomial [22, 50], sabemos que

$$f(x) = p_n(x) + R_n(x), \tag{3.24}$$

em que $R_n(x)$ é o erro na interpolação. Com isso,

$$f(x) - p_n(x) = R_n(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_n)}{(n+1)!} f^{(n+1)}(\xi), \tag{3.25}$$

com $a \leq x \leq b$ e ξ dependendo de x.

Consequentemente, em vista da equação (3.23) e de ξ ser função de x, podemos escrever

$$f(x) - p_n(x) = b_0 \pi_{n+1}(x) \frac{f^{n+1}(x)}{(n+1)!}. \tag{3.26}$$

Como $f(x)$ é um polinômio de grau $\leq 2n + 1$, temos que

$$q(x) = \frac{f^{n+1}(x)}{(n+1)!}, \tag{3.27}$$

é um polinômio de grau $\leq n$. Portanto, podemos escrever

$$f(x) - p_n(x) = b_0 \pi_{n+1}(x) q(x). \tag{3.28}$$

Integrando a equação (3.28) de a até b, com a função peso $\omega(x)$, obtemos

$$\int_a^b \omega(x)[f(x) - p_n(x)]dx = \int_a^b \omega(x) b_0 \pi_{n+1}(x) q(x) dx. \tag{3.29}$$

Como $\pi_{n+1}(x)$ é um polinômio ortogonal (associado a função peso $\omega(x)$ e intervalo$[a,b]$), e $q(x)$ é um polinômio de grau $\leq n$, temos que o lado direito da equação (3.29) é igual a zero. Com isso,

$$\int_a^b \omega(x)[f(x) - p_n(x)]dx = 0 \tag{3.30}$$

$$\int_a^b \omega(x)f(x)dx = \int_a^b \omega(x)p_n(x)dx \tag{3.31}$$

$$= \int_a^b \omega(x)\left[\sum_{k=0}^{n} l_k(x)f(x_k)\right]dx \tag{3.32}$$

$$= \sum_{k=0}^{n} f(x_k)\int_a^b \omega(x)l_k(x)dx \tag{3.33}$$

$$= \sum_{k=0}^{n} w_k f(x_k). \tag{3.34}$$

Portanto, fica provado o resultado. ∎

Na próxima seção deste capítulo, discutiremos uma forma de obtenção das raízes dos polinômios ortogonais, que definem os nós da quadratura gaussiana, através de um problema de autovalores de uma matriz tridiagonal simétrica, juntamente com outras expressões para os pesos de quadratura.

3.2 Nós e Pesos de uma Quadratura Gaussiana

Na seção (3.1), mostramos que as raízes dos polinômios ortogonais são tomadas como nós da quadratura gaussiana. Nesta seção, apresentamos uma forma de obtenção de tais raízes. Isto é feito transformando o problema de obtermos as raízes dos polinômios ortogonais em um problema de autovalores de uma matriz tridiagonal simétrica. Para isso, partimos da relação de recorrência para polinômios

ortonormais, dada pela equação (3.13). A partir dela, para os primeiros n polinômios ortonormais, a relação pode ser reescrita como

$$\begin{cases} x\pi_0^*(x) = & \sqrt{\beta_1}\pi_1^*(x) + \alpha_1\pi_0^*(x) \\ x\pi_1^*(x) = & \sqrt{\beta_2}\pi_2^*(x) + \alpha_2\pi_1^*(x) + \sqrt{\beta_1}\pi_0^*(x) \\ x\pi_2^*(x) = & \sqrt{\beta_3}\pi_3^*(x) + \alpha_3\pi_2^*(x) + \sqrt{\beta_2}\pi_1^*(x) \\ & \vdots \\ x\pi_{n-1}^*(x) = & \sqrt{\beta_n}\pi_n^*(x) + \alpha_n\pi_{n-1}^*(x) + \sqrt{\beta_{n-1}}\pi_{n-2}^*(x), \end{cases} \tag{3.35}$$

ou ainda, em notação matricial, como

$$x\vec{\pi^*}(x) = J_n\vec{\pi^*}(x) + \sqrt{\beta_n}\pi_n^*(x)\vec{e_n}, \tag{3.36}$$

em que $\vec{e_n}$ é a última coluna da matriz identidade de ordem n, $\vec{\pi^*}(x)$ é o vetor

$$\vec{\pi^*}(x) = (\pi_0^*(x)\ \pi_1^*(x)\ \pi_2^*(x)\ \cdots\ \pi_{n-1}^*(x))^T \tag{3.37}$$

e J_n é a matriz de Jacobi de ordem n [37],

$$J_n = \begin{pmatrix} \alpha_1 & \sqrt{\beta_1} & 0 & 0 & \cdots & 0 \\ \sqrt{\beta_1} & \alpha_2 & \sqrt{\beta_2} & 0 & \cdots & 0 \\ 0 & \sqrt{\beta_2} & \alpha_3 & \sqrt{\beta_3} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \sqrt{\beta_{n-2}} & \alpha_{n-1} & \sqrt{\beta_{n-1}} \\ 0 & \cdots & \cdots & 0 & \sqrt{\beta_{n-1}} & \alpha_n \end{pmatrix}. \tag{3.38}$$

Com a relação de recorrência de três termos escrita na forma matricial dada pela equação (3.36), nós temos o seguinte teorema, que relaciona as raízes do polinômio ortogonal de ordem n com o problema de autovalores de uma matriz tridiagonal simétrica [37].

Teorema 3.5. *As raízes $x_1, x_2, \ldots, x_n$ do polinômio ortonormal π_n^* são os autovalores da matriz de Jacobi J_n.*

Demonstração: Se x é uma raiz de π_n^*, da equação (3.36), temos

$$x\overrightarrow{\pi^*}(x) = J_n\overrightarrow{\pi^*}(x). \tag{3.39}$$

Isto mostra que x é um autovalor de J_n com autovetor associado $\overrightarrow{\pi^*}(x)$. ∎

Desta forma, apresentamos um método para a obtenção dos nós da quadratura gaussiana. Com relação aos pesos de quadratura, a demonstração do teorema (3.4) apresenta uma forma de obtenção dos mesmos, através da equação (3.21). Outras fórmulas para a obtenção dos pesos podem ser vistas em Stroud e Secrest [71], por exemplo,

$$w_i = \frac{||\pi_{n-1}||^2}{\pi_n'(x_i)\pi_{n-1}(x_i)}, \tag{3.40}$$

$$w_i = -\frac{||\pi_n||^2}{\pi_n'(x_i)\pi_{n+1}(x_i)}, \tag{3.41}$$

$$w_i = \left[\frac{\pi_0^2(x_i)}{||\pi_0||^2} + \frac{\pi_1^2(x_i)}{||\pi_1||^2} + \cdots + \frac{\pi_{n-1}^2(x_i)}{||\pi_{n-1}||^2}\right]^{-1}, \tag{3.42}$$

$$w_i = \left[(\pi_0^*(x_i))^2 + (\pi_1^*(x_i))^2 + \cdots + (\pi_{n-1}^*(x_i))^2\right]^{-1}. \tag{3.43}$$

Uma outra expressão para os pesos de uma quadratura é dada em relação aos autovetores normalizados da matriz J_n [37, 38], de modo que,

$$w_i = q_{j_1}^2 c_0 \tag{3.44}$$

em que q_{j_1} é a primeira componente do j-ésimo autovetor normalizado da matriz J_n e

$$c_0 = \int_a^b \omega(x)dx. \tag{3.45}$$

Além disso, os pesos da quadratura gaussiana satisfazem a importante propriedade [50, 71]:

Teorema 3.6. *Os pesos da quadratura gaussiana são todos positivos.*

Demonstração: Fixe n. Sejam $\pi_{n+1}(x)$, o polinômio ortogonal de grau $n+1$ associado a função $\omega(x)$ e intervalo $[a,b]$, e x_k, $k=0,1,\ldots,n$, tais que $\pi_{n+1}(x_k)=0$. Defina

$$f_k(x) = \left[\frac{\pi_{n+1}(x)}{x-x_k}\right]^2,\ k=0,1,\ldots,n.$$

Note que, $f_k(x)$ é um polinômio de grau $2n$, e portanto, a fórmula de quadratura gaussiana deve ser exata para este polinômio, isto é,

$$\int_a^b \omega(x) f_k(x)dx = \sum_{i=0}^{n} w_i f_k(x_i).$$

Note também que, devido a forma como $f_k(x)$ foi definida, temos

$$f_k(x_i) = \begin{cases} 0, \text{se } i \neq k, \\ [\pi'_{n+1}(x_i)]^2, \text{se } i = k. \end{cases}$$

Juntando todas as equações, obtemos

$$w_k[\pi'_{n+1}(x_i)]^2 = \sum_{i=0}^{n} w_i f_k(x_i) = \int_a^b \omega(x) f_k(x)dx > 0, \tag{3.46}$$

e o teorema está provado. ∎

Nos casos em que estamos trabalhando com polinômios ortogonais associados a função peso $\omega(x)$ no intervalo $[-a,a]$, com a função peso $\omega(x)$ sendo uma função par, a ordem do problema de autovalores a ser resolvido para a obtenção dos nós de quadratura pode ser reduzida pela metade, como veremos a seguir.

3.2.1 Caso Particular: Polinômios Ortogonais Associados a Função Peso Par $\omega(x)$ no Intervalo $[-a, a]$

O objetivo desta seção é discutirmos um caso particular dos polinômios ortogonais, que são aqueles em que a função peso $\omega(x)$ é uma função par no intervalo simétrico em relação a zero.

Começamos demonstrando uma propriedade importante sobre tais polinômios.

Propriedade 3.4. *Sejam $\omega(x)$ uma função peso par, π_n, $n = 0, 1, 2, \ldots$ polinômios ortogonais associados a função peso $\omega(x)$ no intervalo $[-a, a]$. Então*

$$\pi_n(-x) = (-1)^n \pi_n(x). \tag{3.47}$$

Demonstração: Demonstraremos por indução. Do teorema (3.1), temos que os polinômios ortogonais satisfazem a relação de recorrência dada pela equação (3.8).

Assim, para $n = 0$, temos que $\pi_0(x) = 1$ e o resultado é válido para o caso $n = 0$. Para $n = 1$, temos que

$$\pi_1(x) = x - \frac{\langle x, 1\rangle}{\langle 1, 1\rangle}.$$

Note que,

$$\langle x, 1\rangle = \int_{-a}^{a} \omega(x) x dx = 0,$$

pois $\omega(-x)(-x) = -\omega(x)x$, isto é, o integrando é uma função ímpar. Portanto, o resultado é válido para o caso $n = 1$.

Suponhamos que o resultado seja válido para $i = 0, 1, 2, \ldots, k$, isto é, $\pi_i(-x) = (-1)^i \pi_i(x)$. Vamos provar que o resultado vale para $i = k + 1$. Para $k + 1$,

temos

$$\pi_{k+1}(-x) = (-x - \alpha_k)\pi_k(-x) - \beta_k \pi_{k-1}(-x).$$

Note que,

$$\langle x\pi_k(x), \pi_k(x)\rangle = \int_{-a}^{a} \omega(x) x \pi_k^2(x) dx = 0,$$

uma vez que, devido a hipótese de indução, $\omega(-x)(-x)\pi_k^2(-x) = -\omega(x)x\pi_k^2(x)$, isto é, o integrando é uma função ímpar. Consequentemente, $\alpha_k = 0$.

Assim,

$$\begin{aligned}
\pi_{k+1}(-x) &= -x\pi_k(-x) - \beta_k \pi_{k-1}(-x) \\
&= -x(-1)^k \pi_k(x) - \beta_k (-1)^{k-1} \pi_{k-1}(x) \\
&= (-1)^{k+1} x\pi_k(x) - \beta_k (-1)^{k-1}(-1)^2 \pi_{k-1}(x) \\
&= (-1)^{k+1}[x\pi_k(x) - \beta_k \pi_{k-1}(x)] \\
&= (-1)^{k+1}\pi_{k+1}(x).
\end{aligned}$$

Logo, o resultado é válido para todo n. ∎

Como consequência da propriedade anterior, temos:

1. $\langle x\pi_k(x), \pi_k(x)\rangle = 0, k = 0, 1, 2, \ldots$. Consequentemente, $\alpha_k = 0, k = 1, 2, \ldots$. Note que, para todo k, $\pi_k^2(x)$ é uma função par, e portanto $\omega(x)\pi_k^2(x)x$ é uma função ímpar.

2. Os polinômios ortogonais associados a função peso par $\omega(x)$ e intervalo $[-a, a]$ satisfazem a seguinte relação de recorrência:

$$\begin{cases} \pi_{k+1}(x) = x\pi_k(x) - \beta_k \pi_{k-1}(x), \text{ para } k = 1, 2, \ldots \\ \pi_0(x) = 1 \ , \ \pi_1(x) = x. \end{cases}$$

3. Os polinômios ortonormais associados a função peso par $\omega(x)$ e intervalo $[-a, a]$ satisfazem a seguinte relação de recorrência:

$$\begin{cases} \sqrt{\beta_{k+1}}\pi^*_{k+1}(x) & = x\pi^*_k(x) - \sqrt{\beta_k}\pi^*_{k-1}(x), \text{ para } k = 1, 2, \ldots \\ & \pi^*_0(x) = \frac{1}{||\pi_0||} \ , \ \pi^*_1(x) = \frac{x}{||\pi_1||}. \end{cases} \tag{3.48}$$

Para que possamos reduzir a ordem do problema de autovalores a ser resolvido para obtenção dos nós da quadratura, faremos algumas manipulações algébricas na equação (3.48), que geram os polinômios ortonormais para o caso em que $\omega(x)$ é uma função par e o intervalo em questão é $[-a, a]$.

Começamos multiplicando a equação (3.48) por x, obtendo

$$\sqrt{\beta_{k+1}}x\pi^*_{k+1}(x) = x^2\pi^*_k(x) - \sqrt{\beta_k}\pi^*_{k-1}(x). \tag{3.49}$$

Agora, avaliamos a equação (3.48) em $k+1$ e $k-1$, obtendo

$$\sqrt{\beta_{k+2}}\pi^*_{k+2}(x) = x\pi^*_{k+1}(x) - \sqrt{\beta_{k+1}}\pi^*_k(x), \tag{3.50}$$

e

$$\sqrt{\beta_k}\pi^*_k(x) = x\pi^*_{k-1}(x) - \sqrt{\beta_{k-1}}\pi^*_{k-2}(x). \tag{3.51}$$

Substituindo as equações (3.50) e (3.51) na equação (3.49), temos

$$x^2\pi^*_k(x) = \sqrt{\beta_{k+1}\beta_{k+2}}\pi^*_{k+2}(x) + (\beta_{k+1} + \beta_k)\pi^*_k(x) + \sqrt{\beta_k\beta_{k-1}}\pi^*_{k-2}(x). \tag{3.52}$$

A partir da equação (3.52), podemos transformar o problema de encontrar as raízes do polinômio ortogonal em um problema de autovalores de uma matriz tridiagonal simétrica de ordem reduzida. Por exemplo, se quisermos determinar as

raízes de $\pi_6(x)$, a partir da equação (3.52), temos

$$\begin{cases} x^2\pi_0^* = \sqrt{\beta_1\beta_2}\pi_2^*(x) + \beta_1\pi_0^* \\ x^2\pi_2^* = \sqrt{\beta_3\beta_4}\pi_4^*(x) + (\beta_3+\beta_2)\pi_2^* + \sqrt{\beta_2\beta_1}\pi_0^*(x) \\ x^2\pi_4^* = (\beta_5+\beta_4)\pi_4^* + \sqrt{\beta_4\beta_3}\pi_2^*(x) \end{cases} \tag{3.53}$$

já que $\pi_6^*(x) = 0$.

Escrevendo em notação matricial,

$$x^2\overrightarrow{\pi^*}(x) = H\overrightarrow{\pi^*}(x), \tag{3.54}$$

com

$$H = \begin{pmatrix} \beta_1 & \sqrt{\beta_1\beta_2} & 0 \\ \sqrt{\beta_1\beta_2} & \beta_2+\beta_3 & \sqrt{\beta_3\beta_4} \\ 0 & \sqrt{\beta_3\beta_4} & \beta_4+\beta_5 \end{pmatrix} \tag{3.55}$$

e

$$\overrightarrow{\pi^*}(x) = (\pi_0^*(x)\ \pi_2^*(x)\ \pi_4^*(x))^T \tag{3.56}$$

Por fim, para os casos em que n é ímpar, da equação (3.52), obtemos que $x = 0$ sempre será uma raiz de π_n^*. De fato, podemos provar isso por indução. Para $n = 1$, temos que $\pi_1^*(x) = \frac{x}{||\pi_1||}$, e portanto, o resultado segue. Suponhamos válido para $i = 1, 3, 5, \ldots, k-2, k$, e provemos para $i = k+2$. De fato, avaliando a equação (3.52) em $x = 0$ para $i = k+2$, temos

$$0 = 0^2\pi_k^*(0) \tag{3.57}$$

$$= \sqrt{\beta_{k+1}\beta_{k+2}}\pi_{k+2}^*(0) + (\beta_{k+1}+\beta_k)\pi_k^*(0) + \sqrt{\beta_k\beta_{k-1}}\pi_{k-2}^*(0) \tag{3.58}$$

$$= \sqrt{\beta_{k+1}\beta_{k+2}}\pi_{k+2}^*(0) \tag{3.59}$$

$$= \pi_{k+2}^*(0), \tag{3.60}$$

devido a hipótese de indução. Assim, o resultado está provado.

Na seção seguinte, apresentamos estimativas para o erro de truncamento.

3.3 Estimativa para o Erro de Truncamento

Iniciamos esta seção definindo o erro truncamento de uma fórmula de quadratura gaussiana [30].

Definição 3.4. *Seja*

$$\sum_{k=0}^{n} w_k f(x_k) \tag{3.61}$$

uma fórmula de quadratura gaussiana, tal que

$$\int_a^b \omega(x) f(x) dx \approx \sum_{k=0}^{n} w_k f(x_k). \tag{3.62}$$

Definimos o erro de truncamento da fórmula de quadratura, denotado por $E_n(f)$, como:

$$E_n(f) = \int_a^b \omega(x) f(x) dx - \sum_{k=0}^{n} w_k f(x_k). \tag{3.63}$$

Para estimar o erro de truncamento da quadratura gaussiana, assumimos que $f(x)$ pertence ao espaço de funções $\mathcal{C}_Z^r$ (Brass e Petras [14] se referem a isto como "co-observação") tal que

$$\mathcal{C}_Z^r := \{f \in C[a,b] : \sup \operatorname{ess}_{a \leq x \leq b} |f^{(r)}(x)| \leq Z\},$$

em que $f^{(r)}$ é a r-ésima derivada generalizada de $f(x)$. Nosso próximo teorema tratará sobre o erro de truncamento da quadratura gaussiana para o caso em que f é suficientemente suave ($r \geq 2n+2$).

Teorema 3.7. *Seja $f \in C^{2n+2}[a,b]$. Então, o erro para a fórmula de quadratura gaussiana de ordem $n+1$ é dado por*

$$\int_a^b \omega(x)f(x)dx - \sum_{k=0}^{n} w_k f(x_k) = \frac{f^{2n+2}(\xi)}{(2n+2)!}\int_a^b \omega(x)[\pi_{n+1}(x)]^2 dx, \tag{3.64}$$

para algum $\xi \in [a,b]$.

Demonstração: A demonstração é feita de acordo com Kress [50]. Seja $H_n f \in P_{2n+1}$ o polinômio interpolador de hermite para f [50]. Uma vez que $(H_n f)(x_k) = f(x_k)$, $k = 0, 1, \ldots, n$, o erro de truncamento

$$E_n(f) := \int_a^b \omega(x)f(x)dx - \sum_{k=0}^{n} w_k f(x_k) \tag{3.65}$$

pode ser escrito como

$$E_n(f) = \int_a^b \omega(x)[f(x) - (H_n f)(x)]dx. \tag{3.66}$$

Usando o teorema do valor médio para integrais [59], temos

$$E_n(f) = \frac{f(z) - (H_n f)(z)}{[\pi_{n+1}(z)]^2}\int_a^b \omega(x)[\pi_{n+1}(x)]^2 dx, \tag{3.67}$$

para algum $z \in [a,b]$. Por sua vez, o erro na interpolação de Hermite é dado por [50]

$$f(z) - (H_n f)(z) = \frac{f^{2n+2}(\xi)}{(2n+2)!}\prod_{j=0}^{n}(z - z_j)^2, x \in [a,b], \tag{3.68}$$

para algum $\xi \in [a,b]$ dependendo de z. Com isso, concluímos que o erro de truncamento da quadratura gaussiana é

$$E_n(f) = \frac{f^{2n+2}(\xi)}{(2n+2)!}\int_a^b \omega(x)[\pi_{n+1}(x)]^2 dx, \tag{3.69}$$

para algum $\xi \in [a,b]$. ∎

Note que, a estimativa do erro de truncamento para a regra de quadratura gaussiana exige um grande suavidade da função. Contudo, segundo Brass e Petras [14], no pior caso, o erro de truncamento para uma regra de quadratura com n pontos e todas as funções $f \subset \mathcal{C}_Z^r$ converge como:

$$E_n(f) = \mathcal{O}(n^{-r}). \tag{3.70}$$

Além disso, segundo Kütz [51], se $f(x) \subset \mathcal{C}_Z^r$ possui uma singularidade no interior do intervalo de integração, então o erro de truncamento de uma regra de quadratura gaussiana com n pontos converge como:

$$E_n(f) = \mathcal{O}(n^{-r-1}). \tag{3.71}$$

Este resultados serão importantes na estimativa para o erro de truncamento das quadraturas produto. Nas seções seguintes, discutimos dois casos particulares de quadraturas gaussianas: Quadratura de Gauss-Chebyshev e Quadratura de Gauss-Legendre.

3.4 Quadratura de Gauss-Chebyshev

Chamamos de quadratura de Gauss-Chebyshev [30] a fórmula de quadratura

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx \approx \sum_{k=1}^{n} w_k^c f(x_k) \tag{3.72}$$

com n nós capaz de integrar exatamente todos os polinômios $q \in P_{2n-1}$, isto é, na definição (3.1), estamos considerando como função peso

$$\omega(x) = \frac{1}{\sqrt{1-x^2}}, \tag{3.73}$$

$a = -1$ e $b = 1$. A quadratura recebe este nome pois os polinômios ortogonais associados a esta função peso e intervalo são os polinômios de Chebyshev de primeira espécie. Com isso, da discussão feita nas seções anteriores, temos que os nós da quadratura de Gauss-Chebyshev são as raízes dos polinômios de Chebyshev de 1º espécie.

Definimos os polinômios de Chebyshev de 1º espécie, de acordo com Golub e Meurant [37], como:

Definição 3.5. *Os polinômios de Chebyshev de* 1^o *espécie,* $T_n(x)$*, são definidos como*

$$T_n(x) = \cos(n \arccos(x)). \tag{3.74}$$

Tais polinômios gozam das seguintes propriedades [15, 22, 23, 50]:

- $|T_n(x)| \leq 1, \forall x \in \mathbb{R}$;
- As raízes de $T_n(x)$ são $x_k = \cos(\frac{(2k-1)\pi}{2n}), k = 1, 2, \ldots, n$, e estão contidas no intervalo $[-1, 1]$;
- Satisfazem a seguinte condição de ortogonalidade:

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x) T_m(x) dx = \begin{cases} 0, m \neq n \\ \frac{\pi}{2}, m = n \neq 0 \\ \pi, m = n = 0 \end{cases} ;$$

- Satisfazem a seguinte relação de recorrência:

$$\begin{cases} T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), n \geq 1 \\ T_0(x) = 1, T_1(x) = x \end{cases} .$$

Por fim, os pesos da quadratura de Gauss-Chebyshev, w_k^c, são todos iguais a π/n [23, 50]. Assim, podemos escrever a expressão da quadratura de Gauss-Chebyshev de ordem n, incluindo o termo de erro de truncamento, como [23]

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx = \frac{\pi}{n} \sum_{k=1}^{n} f(x_k) + \frac{\pi}{2^{2n-1}(2n)!} f^{(2n)}(\xi), \tag{3.75}$$

em que $x_k = \cos(\frac{(2k-1)\pi}{2n})$, $k = 1, 2, \ldots, n$ e $\xi \in (-1, 1)$.

3.5 Quadratura de Gauss-Legendre

Chamamos de quadratura de Gauss-Legendre [30] a fórmula de quadratura

$$\int_{-1}^{1} f(x)dx \approx \sum_{k=1}^{n} w_k^l f(x_k) \tag{3.76}$$

com n nós capaz de integrar exatamente todos os polinômos $q \in P_{2n-1}$, isto é, na definição (3.1), estamos considerando como função peso

$$\omega(x) = 1, \tag{3.77}$$

$a = -1$ e $b = 1$. A quadratura recebe este nome pois os polinômios ortogonais associados a esta função peso e intervalo são os polinômios de Legendre. Denotamos o polinômio de Legendre de grau l por $P_l(x)$. Com isso, da discussão feita nas seções anteriores, temos que os nós da quadratura de Gauss-Legendre são as raízes dos polinômios de Legendre.

Os polinômios de Legendre gozam das seguintes propriedades [15, 22, 23]:

- $|P_n(x)| \leq 1, \forall x \in [-1, 1]$;

- $$\int_{-1}^{1} P_n^2(x)dx = \frac{2}{2n+1}$$

- Satisfazem a seguinte relação de recorrência:

$$\begin{cases} (n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) \\ P_0(x) = 1, P_1(x) = x. \end{cases} \tag{3.78}$$

Ao contrário do que acontece na quadratura de Gauss-Chebyshev, em que os nós são facilmente gerados e todos os pesos de quadratura são iguais a π/n, o mesmo não ocorre na quadratura de Gauss-Legendre. Desta forma, da discussão feita na seção 3.2, obtemos os nós da quadratura de Gauss-Legendre resolvendo um problema de autovalores de uma matriz tridiagonal simétrica. Em particular, para o caso em que n é par, por meio de manipulações algébricas da equação (3.78), podemos reescrevê-la como [7]

$$x^2 P_n^*(x) = \sqrt{a_{n+2}a_{n+1}}P_{n+2}^*(x) + (a_{n+1} + a_n)P_n^*(x) + \sqrt{a_n a_{n-1}}P_{n-2}^*(x) \tag{3.79}$$

com $P_n(x) = (2/h_n)^{1/2}P_n^*(x)$, $a_n = n^2/(h_n h_{n-1})$ e $h_n = 2n+1$ para $n = 0, 2, ...$

Através da equação (3.79), obtemos os nós da quadratura de Gauss-Legendre de ordem n (n par) resolvendo um problema de autovalores de uma matriz tridiagonal simétrica de ordem reduzida. Por exemplo, se quisermos obter as raízes de $P_8(x)$, utilizando a equação (3.79), basta resolvermos o seguinte problema de autovalores

$$H\,\vec{v} = x^2\,\vec{v},$$

em que

$$H = \begin{pmatrix} a_1 & \sqrt{a_1 a_2} & 0 & 0 \\ \sqrt{a_1 a_2} & a_2 + a_3 & \sqrt{a_3 a_4} & 0 \\ 0 & \sqrt{a_3 a_4} & a_4 + a_5 & \sqrt{a_5 a_6} \\ 0 & 0 & \sqrt{a_5 a_6} & a_6 + a_7 \end{pmatrix}$$

e

$$\vec{v} = (P_0^*(x)\ P_2^*(x)\ P_4^*(x)\ P_6^*(x))^T,$$

uma vez que $P_8^*(x) = 0$, e cujos autovalores, $\lambda = x^2$, são o quadrado das raízes positivas do polinômio de Legendre grau n (notando que as raízes aparecem aos pares $\pm$). Já os pesos da quadratura de Gauss-Legendre, w_k^l, podem ser obtidos através de uma das fórmulas citadas na seção (3.2).

Por fim, podemos escrever a expressão da quadratura de Gauss-Legendre de ordem n, incluindo o termo de erro de truncamento, como [23]

$$\int_{-1}^{1} f(x)dx = \sum_{k=1}^{n} w_k^l f(x_k) + \frac{2^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi), \tag{3.80}$$

em que $\xi \in (-1, 1)$.

No próximo capítulo utilizaremos os conceitos deste capítulo para a obtenção de quadraturas produto para a esfera unitária.

4 QUADRATURAS NUMÉRICAS NA ESFERA UNITÁRIA

Neste capítulo estudamos os esquemas de integração na esfera unitária chamados: Legendre-Chebyshev Quadrangular ($P_N T_N$) [15, 60, 73], Legendre-Chebyshev Triangular ($P_N T_N S_N$)[15, 60, 73] e *Quadruple Range* (QR) [1, 2, 70]. Também apresentamos alguns resultados associados ao erro de truncamento destas quadraturas multidimensionais.

Como mencionado anteriormente, o motivo para o estudo destes esquemas de quadraturas está relacionado ao método das ordenadas discretas e ao esquema clássico utilizado na área, chamado de *Level Symmetric* (LQ_N) [57]. Tal esquema de quadratura apresenta uma limitação quanto a ordem utilizada, sendo que, para ordens $N > 20$ (o que representa 55 direções discretas por octante da esfera unitária), começam a surgir pesos de quadratura negativos, implicando em soluções fisicamente impossíveis [54]. Contudo, os esquemas $P_N T_N$, $P_N T_N S_N$ e QR contornam essa limitação no número de direções, uma vez que todos os seus pesos de quadratura são positivos. Desta forma, esperamos que, com um número maior de direções discretas utilizadas, a solução em ordenadas discretas melhor se aproxime da solução exata do problema, bem como a diminuição do efeito raio.

4.1 Quadraturas Legendre-Chebyshev

Nesta seção, apresentamos dois conjuntos de quadraturas gerados através do produto da quadratura unidimensional de Gauss-Legendre e a quadratura uni-

dimensional de Gauss-Chebyshev. Tais conjuntos se diferenciam pela forma como relacionamos as ordens das quadraturas unidimensionais utilizadas, bem como a forma como são definidos os pesos da quadratura produto.

4.1.1 Quadratura Legendre-Chebyshev Quadrangular (P_NT_N)

Na quadratura de P_NT_N, tanto a ordem da quadratura unidimensional de Gauss-Legendre quanto a ordem da quadratura de Gauss-Chebyshev utilizadas são iguais [15, 73]. Para sua construção, cada nível polar, ξ_i, é definido como a i-ésima raiz do polinômio de Legendre de grau N, P_N . Uma vez determinados os níveis polares, determinamos o conjunto discreto de ângulos azimutais através da relação [15, 73]

$$\varpi_j = \frac{\pi}{2}\left(1 - \frac{N-2j+1}{N}\right), \tag{4.1}$$

em que $j = 1, 2, \ldots, N$, e N é a ordem da quadratura de Gauss-Legendre. Note que, o conjunto discreto de ângulos azimutais obtidos pela equação (4.1) determinam as raízes do polinômio de Chebyshev de 1ª espécie de grau N, T_N, uma vez que

$$T_N(\cos(\varpi_j)) = 0. \tag{4.2}$$

Obtidos os valores de ϖ_j, definimos o conjunto de pontos da quadratura produto P_NT_N por [15, 73]

$$\mu_{i,j} = \cos(\varpi_j)\sqrt{1-\xi_i^2} \tag{4.3}$$

e

$$\eta_{i,j} = \sqrt{1-\xi_i^2-\mu_{i,j}^2}, \tag{4.4}$$

com $i = 1, \ldots, N/2$ e $j = 1, \ldots, N$.

Por sua vez, os pesos da quadratura $P_N T_N$ são dados em relação aos pesos w_i^l da quadratura de Gauss-Legendre, como [15]

$$W_{i,j} = \frac{\pi w_i^l}{N}, \tag{4.5}$$

de modo que, para cada nível polar ξ_i, as direções correspondentes possuem o mesmo peso, e sendo que, neste trabalho, a soma dos pesos deve ser 4π. Também estamos considerando a ordem N das quadraturas unidimensionais utilizadas na construção da quadratura $P_N T_N$ sendo um número natural par.

A quadratura $P_N T_N$ assume um padrão quadrangular de disposição das direções na esfera unitária, conforme pode ser observado na figura 4.1, e resulta no total de $M = N^2/4$ direções discretas por octante da esfera unitária.

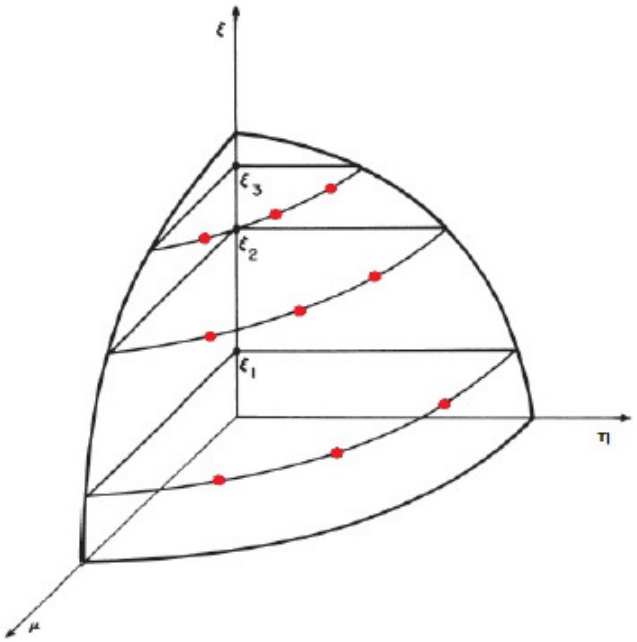
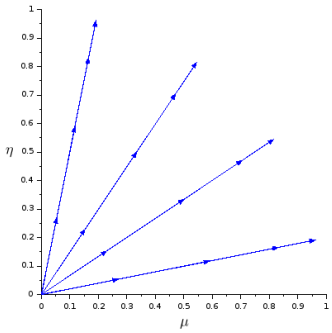
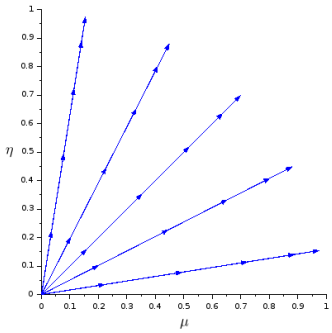


Figura 4.1 Distribuição de direções na quadratura $P_6 T_6$ (Fonte: Tres [73])

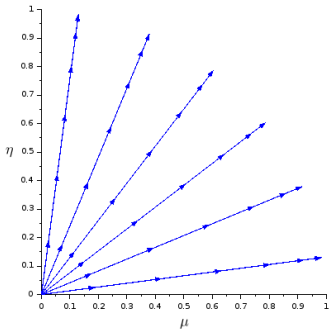
Por fim, devido a forma como este acoplamento é realizado, quando as direções são projetadas no plano $\mu\eta$, observamos que há direções que ficam sobrepostas, como pode ser observado na figura 4.2. Este comportamento é reflexo do fato de todos os níves polares utilizarem a mesma ordem de quadratura azimutal.



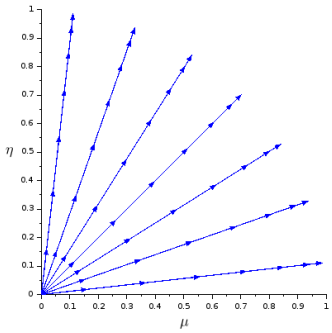
(a) P_8T_8



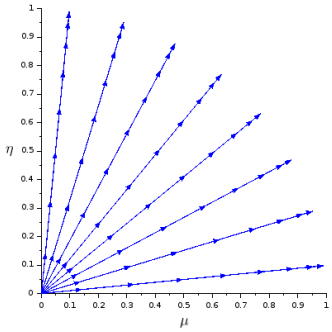
(b) $P_{10}T_{10}$



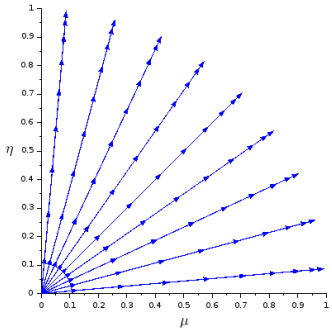
(c) $P_{12}T_{12}$



(d) $P_{14}T_{14}$



(e) $P_{16}T_{16}$



(f) $P_{18}T_{18}$

Figura 4.2 Distribuição das direções discretas da quadratura P_NT_N no plano $\mu\eta$

4.1.2 Quadratura Legendre-Chebyshev Triangular ($P_N T_N S_N$)

Assim como na quadratura $P_N T_N$, na quadratura $P_N T_N S_N$ também estamos considerando que a ordem N da quadratura de Gauss-Legendre utilizada é um número natural par. Novamente, cada nível polar, ξ_i, é definido como a i-ésima raiz do polinômio de Legendre de grau N, P_N, e, uma vez definidos os ξ_i níveis polares, a discretização da variável azimutal é feita de forma que, para o primeiro nível polar, ξ_1, utilizamos a quadratura de Gauss-Chebyshev de ordem N; no segundo nível polar, ξ_2, utilizamos a quadratura de Gauss-Chebyshev de ordem $N-2$; no i-ésimo nível polar, ξ_i, utilizamos a quadratura Gauss-Chebyshev de ordem $N-2i+2$. Como estamos considerando que a ordem N da quadratura de Gauss-Legendre deve ser par, temos que os níveis polares aparecem aos pares $\pm$, e portanto temos $N/2$ níveis polares positivos. Além disso, estamos considerando o primeiro nível polar, ξ_1, aquele mais próximo do centro da esfera unitária, enquanto o último nível polar, $\xi_{N/2}$, é o mais próximo da borda da esfera unitária.

Desta forma, na construção da quadratura $P_N T_N S_N$, determinamos o conjunto discreto de ângulos azimutais para cada nível ξ_i através da relação [15, 73]

$$\varpi_{i,j} = \frac{\pi}{2}\left(1 - \frac{N-2j-2i+3}{N-2i+2}\right), \tag{4.6}$$

em que $i = 1, 2, \ldots, N/2$, $j = 1, 2, \ldots, N-2i+2$, e N é a ordem da quadratura de Gauss-Legendre. Como no i-ésimo nível polar é utilizada a quadratura de Gauss-Chebyshev de ordem $N-2i+2$, para cada valor de i, o conjunto discreto de ângulos azimutais obtidos pela equação (4.6) determinam as raízes do polinômio de Chebyshev de 1ª espécie de grau $N-2i+2$, uma vez que

$$T_{N-2i+2}(\cos(\varpi_{i,j})) = 0. \tag{4.7}$$

Obtidos os valores de $\varpi_{i,j}$, definimos o conjunto de pontos da quadratura produto $P_N T_N S_N$ por [15, 73]

$$\mu_{i,j} = \cos(\varpi_{i,j})\sqrt{1-\xi_i^2} \tag{4.8}$$

e

$$\eta_{i,j} = \sqrt{1-\xi_i^2-\mu_{i,j}^2}, \tag{4.9}$$

com $i = 1, \ldots, N/2$ e $j = 1, \ldots, N-2i+2$.

Por sua vez, os pesos da quadratura $P_N T_N S_N$ são dados em relação aos pesos w_i^l da quadratura de Gauss-Legendre como [15]

$$W_{i,j} = \frac{\pi w_i^l}{N-2i+2}, \tag{4.10}$$

de modo que, neste trabalho, a soma dos pesos deve ser 4π.

Devido a forma como são combinadas as ordens das quadraturas unidimensionais, o esquema de quadratura $P_N T_N S_N$ assume um padrão triangular de disposição das direções na esfera unitária, como pode ser observado na figura 4.3, e possui o total de $M = N(N+2)/8$ direções discretas por octante.

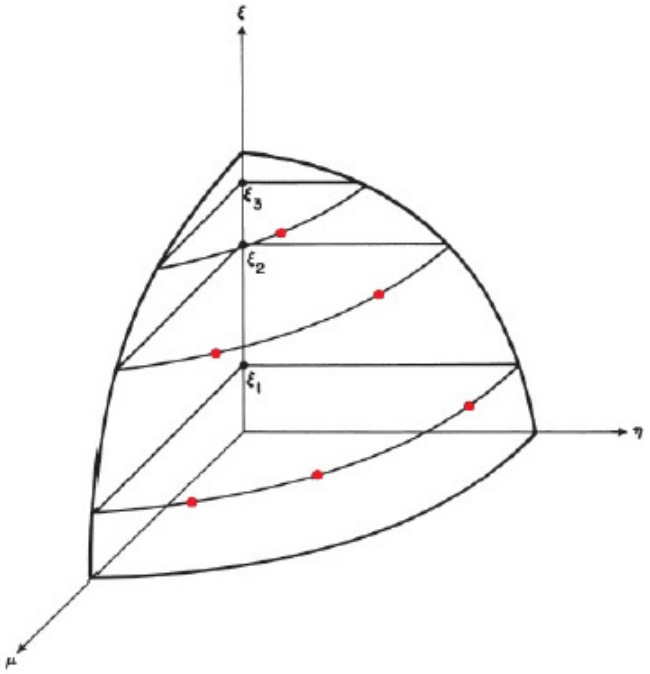
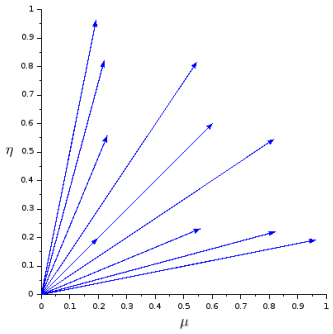


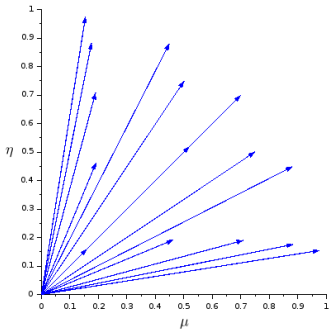
Figura 4.3 Distribuição de direções na quadratura $P_6T_6S_6$ (Fonte: Tres [73])

Por fim, quando as direções são projetadas no plano $\mu\eta$, em comparação com a quadratura P_NT_N, observamos que há um número menor de direções que ficam sobrepostas, como pode ser observado na figura 4.4. Este comportamento decorre do fato de não utilizarmos a mesma ordem azimutal em todos os níveis polares.

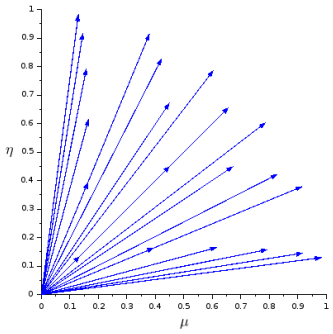
Na seção seguinte, apresentamos os aspectos computacionais para a implementação destes esquemas de quadratura.



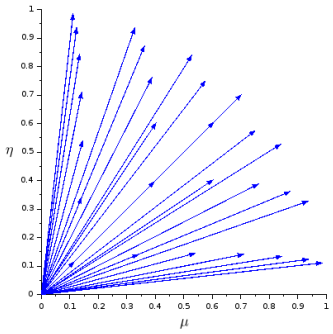
(a) $P_8T_8S_8$



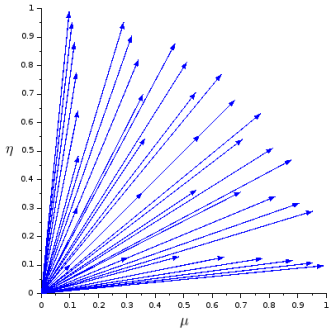
(b) $P_{10}T_{10}S_{10}$



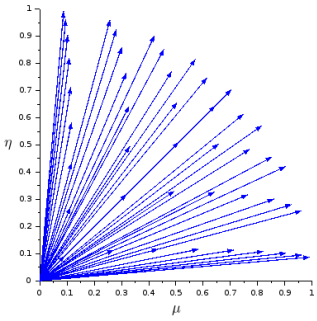
(c) $P_{12}T_{12}S_{12}$



(d) $P_{14}T_{14}S_{14}$



(e) $P_{16}T_{16}S_{16}$



(f) $P_{18}T_{18}S_{18}$

Figura 4.4 Distribuição das direções discretas da quadratura $P_NT_NS_N$ no plano $\mu\eta$

4.1.3 Aspectos computacionais dos esquemas $P_N T_N$ e $P_N T_N S_N$

Implementamos em linguagem Fortran 95 [45] códigos para a geração dos esquemas de quadratura $P_N T_N$ e $P_N T_N S_N$ em precisão simples, dupla e quádrupla. Utilizamos a precisão quádrupla, pois estamos interessados em gerar os esquemas de quadratura com máxima precisão possível.

Como foi visto na seção 3.4, os nós da quadratura de Gauss-Chebyshev de ordem N são obtidos através de

$$x_i = \cos\left(\frac{(2i-1)\pi}{2N}\right), \tag{4.11}$$

para $i = 1, \ldots, N$, e seus pesos de quadratura são todos iguais a π/N. Por sua vez, os nós da quadratura de Gauss-Legendre são obtidos resolvendo um problema de autovalores de uma matriz simétrica tridiagonal, conforme foi discutido na seção 3.5, e seus pesos de quadratura são obtidos através da equação (3.43).

Na geração destes esquemas de quadratura em precisão simples ou dupla, o problema de autovalores que surge na obtenção dos nós da quadratura de Gauss-Legendre pode ser resolvido através das rotinas *SSTEVD* [3], *DSTEVD* [3] e *iter_qr* [65]. As três rotinas são específicas para a resolução do problema de autovalores para matrizes tridiagonais simétricas, sendo que as duas primeiras fazem parte da biblioteca LAPACK [3] e a última é uma implementação da iteração QR implícita com shift de Wilkinson [21, 36] feita por nós. Como a biblioteca LAPACK não está implementada em precisão quádrupla, o problema de autovalores para precisão quádrupla somente pode ser resolvido pela rotina *iter_qr*.

Nas tabelas 4.1 e 4.2, apresentamos o tempo médio em segundos de 1000 execuções para a geração das quadraturas $P_N T_N$ e $P_N T_N S_N$, resolvendo o problema

de autovalores através das rotinas da biblioteca LAPACK e da rotina *iter_ qr*, para precisão simples, dupla e quádrupla. O computador utilizado para tal foi um notebook HP Pavilion 15-b102sp Sleekbook, munido de um processor Intel Core i5-3337U, com clock de $1,8$ GHz (chegando até $2,7$ GHz com o turbo boost), com 3 Mb de memória cache L3, 6 GB de memória RAM DDR3 com clock de 1600 MHz, utilizando o sistema operacional Linux Mint 19.2 "Tina" - Cinnamon de 64 bits. Em relação ao compilador para Fortran 95, utilizamos o compilador gfortran 7.4.0 [35], e para a medição do tempo, utilizamos o intrínseco do Fortran 95 DATE_AND_TIME [18], que mede o tempo com precisão de milisegundos.

Tabela 4.1 Tempo médio de execução (em segundos) para geração da quadratura $P_N T_N$

Ordem	**Precisão Simples**		**Precisão Dupla**		**Precisão Quádrupla**
	SSTEVD	*iter_ qr*	*DSTEVD*	*iter_ qr*	*iter_ qr*
10	1.73E-04	2.00E-06	8.00E-06	5.00E-06	1.40E-05
20	6.00E-06	1.00E-05	1.60E-05	2.20E-05	3.20E-05
30	1.40E-05	1.50E-05	3.30E-05	4.20E-05	7.80E-05
40	2.50E-05	2.50E-05	3.60E-05	5.00E-05	7.00E-05
50	3.60E-05	3.90E-05	5.70E-05	6.20E-05	1.07E-04
60	5.10E-05	5.80E-05	7.50E-05	8.60E-05	1.42E-04
70	7.30E-05	7.80E-05	1.04E-04	1.15E-04	1.95E-04
80	9.20E-05	1.02E-04	1.41E-04	1.50E-04	2.49E-04
90	1.13E-04	1.26E-05	1.71E-04	1.91E-04	3.00E-04
100	1.39E-04	1.52E-05	2.19E-04	2.35E-04	3.87E-04

Tabela 4.2 Tempo médio de execução (em segundos) para geração da quadratura $P_N T_N S_N$

Ordem	**Precisão Simples**		**Precisão Dupla**		**Precisão Quádrupla**
	SSTEVD	**iter_qr**	**DSTEVD**	**iter_qr**	**iter_qr**
10	4.00E-06	7.00E-06	7.00E-06	7.00E-06	1.30E-05
20	1.30E-05	1.60E-05	2.00E-05	2.00E-05	4.90E-05
30	2.70E-05	3.50E-05	4.20E-05	4.50E-05	6.80E-05
40	2.80E-05	4.60E-05	4.10E-05	4.80E-05	7.30E-05
50	4.10E-05	4.50E-05	6.20E-05	7.00E-05	1.07E-04
60	5.70E-05	6.30E-05	8.90E-05	9.90E-05	1.56E-04
70	7.90E-05	8.50E-05	1.19E-04	1.32E-04	2.12E-04
80	9.90E-05	1.10E-04	1.54E-04	1.72E-04	2.63E-04
90	1.20E-04	1.40E-04	2.01E-04	2.16E-04	3.38E-04
100	1.57E-04	1.73E-04	2.45E-04	2.65E-04	4.05E-04

Na seção seguinte, apresentamos o esquema de quadratura *Quadruple Range*.

4.2 Quadratura *Quadruple Range* (QR)

A quadratura *Quadruple Range* (QR) é uma quadratura produto proposta por Abu-Shumays [1] na década de 70, desenvolvida para o tratamento de problemas bidimensionais de transporte. Esta quadratura se diferencia das anteriores, pois a sua construção é feita somente para o octante principal da esfera unitária [1, 2], isto é, consideramos o intervalo de definição para o ângulo azimutal $\phi \in [0, \pi/2]$

e para o ângulo polar o intervalo $\theta \in [0, \pi/2]$ [2]. Uma vez definidas M direções discretas e seus respectivos pesos de quadratura para o octante principal, eles são estendidos para os demais intervalos angulares por simetria [1, 2].

A proposta inicial de Abu-Shumays para a construção da quadratura QR envolve a resolução de sistemas não lineares mal-condicionados [1, 2], e devido a isso, somente esquemas de baixa ordem foram gerados numericamente (para a quadratura azimutal, $N_\phi \leq 22$; para a quadratura polar, $N_\theta \leq 10$). [1, 2]. Para contornar o problema do mal condicionamento destes sistemas não lineares, Spence [70] propôs uma abordagem utilizando polinômios ortogonais não clássicos.

A seguir, apresentamos os aspectos teóricos e computacionais para o desenvolvimento das quadraturas QR, tanto para a abordagem via sistemas não lineares proposta por Abu-Shumays, quanto para a abordagem via polinômios ortogonais não clássicos proposta por Spence, e por fim, discutimos o acoplamento das quadraturas unidimensionais.

4.2.1 Quadratura QR via sistemas não lineares

Iniciamos o estudo desta abordagem pelo tratamento da variável polar. Para construírmos esta quadratura, escrevemos a integral sobre o ângulo polar dada pela equação (2.35), de acordo com Abu-Shumays [1], como

$$\int_0^\pi \sin^{l+m}\theta \sin\theta d\theta = 2\int_0^{\pi/2} \sin^{l+m}\theta \sin\theta d\theta \tag{4.12}$$

$$= 2\int_0^1 x^k \frac{xdx}{\sqrt{1-x^2}} \tag{4.13}$$

em que foram feitas as mudanças de variáveis $\sin(\theta) = x$, e $k = l + m$.

A integral dada pela equação (4.13) é então aproximada pela quadratura de Gauss-Christoffel [34], em que

$$\int_0^1 f(x)\omega(x)dx \approx \sum_{i=1}^{N_\theta} w_i^p f(x_i), \tag{4.14}$$

com $f(x) = x^k$ e $\omega(x) = x/\sqrt{1-x^2}$.

Desta forma, geramos os nós, x_i, e pesos, w_i^p, da quadratura de ordem N_θ a partir de (4.14) tornando tal equação exata para todos os polinômios x^j, com $j = 0, 1, \ldots, 2N_\theta - 1$ [1, 2, 4, 50]. Por exemplo, para obtermos os nós e pesos da quadratura de ordem $N_\theta = 2$, resolvemos o sistema não linear

$$\begin{cases} w_1^p + w_2^p = \int_0^1 \frac{xdx}{\sqrt{1-x^2}} \\ w_1^p x_1 + w_2^p x_2 = \int_0^1 x\frac{xdx}{\sqrt{1-x^2}} \\ w_1^p x_1^2 + w_2^p x_2^2 = \int_0^1 x^2\frac{xdx}{\sqrt{1-x^2}} \\ w_1^p x_1^3 + w_2^p x_2^3 = \int_0^1 x^3\frac{xdx}{\sqrt{1-x^2}}. \end{cases} \tag{4.15}$$

Em relação à variável azimutal, procuramos uma quadratura para aproximar a integral [1, 2]

$$\int_0^{\pi/2} \cos^l(\phi)\sin^m(\phi)d\phi \approx \sum_{j=1}^{N_\phi} w_j^a \cos^l(\phi_j)\sin^m(\phi_j), \tag{4.16}$$

em que escolhemos ângulos azimutais com simetria em relação a $\phi = \pi/4$ [1]. Segundo Abu-Shumays [1], para que haja esta simetria, a quadratura para a variável azimutal de ordem N_ϕ deve satisfazer as seguintes condições:

- **Caso N_ϕ seja par**: Para $j = 1, \ldots, N_\phi/2$, temos que $\phi_j < \pi/4$ e os pesos w_j^a necessitam serem calculados. As direções $\phi_{\frac{N_\phi}{2}+1}, \phi_{\frac{N_\phi}{2}+2}, \ldots, \phi_{N_\phi}$ e os

respectivos pesos devem satisfazer:

$$\phi_{N_\phi+1-j} = \frac{\pi}{2} - \phi_j, \tag{4.17}$$

$$w_j^a = w_{N_\phi+1-j}^a. \tag{4.18}$$

- **Caso N_ϕ seja ímpar**: Para $j = 1, \ldots, (N_\phi - 1)/2$, temos que $\phi_j < \pi/4$ e os pesos w_j^a necessitam serem calculados. As direções $\phi_{\frac{N_\phi-1}{2}+2}, \phi_{\frac{N_\phi-1}{2}+3}, \ldots, \phi_{N_\phi}$ e os respectivos pesos devem satisfazer:

$$\phi_{N_\phi+1-j} = \frac{\pi}{2} - \phi_j, \tag{4.19}$$

$$w_j^a = w_{N_\phi+1-j}^a. \tag{4.20}$$

O caso $j = (N_\phi + 1)/2$, o peso de quadratura, $w_{(N_\phi+1)/2}^a$, deve ser calculado e

$$\phi_{\frac{N_\phi+1}{2}} = \frac{\pi}{4}. \tag{4.21}$$

Exemplificamos isto a seguir, para $N_\phi = 4$ e $N_\phi = 5$:

- $N_\phi = 4$**:** Neste caso, temos:

$$\phi_1 \leq \frac{\pi}{4}\ ,\ \phi_2 \leq \frac{\pi}{4}; \tag{4.22}$$

$$\phi_3 = \frac{\pi}{2} - \phi_2\ ,\ \phi_4 = \frac{\pi}{2} - \phi_1; \tag{4.23}$$

$$w_3^a = w_2^a \text{ e } w_4^a = w_1^a. \tag{4.24}$$

- $N_\phi = 5$**:** Neste caso, temos:

$$\phi_1 \leq \frac{\pi}{4}\ ,\ \phi_2 \leq \frac{\pi}{4}; \tag{4.25}$$

$$\phi_4 = \frac{\pi}{2} - \phi_2\ ,\ \phi_5 = \frac{\pi}{2} - \phi_1; \tag{4.26}$$

$$w_4^a = w_2^a \ , \ w_5^a = w_1^a; \tag{4.27}$$

$$\phi_3 = \frac{\pi}{4} \text{ e } w_3^a \text{ precisa ser calculado.} \tag{4.28}$$

A vantagem em escolher ângulos azimutais com simetria em relação a $\phi = \pi/4$ é que reduz pela metade o número de nós e pesos da quadratura que precisam ser computados [2]. Além disso, devido as condições de simetria, temos que os integrandos da equação (4.16) formam um conjunto linearmente dependente [1, 2], e como consequência, consideramos somente os seguintes integrandos na construção do sistema não linear necessário para a obtenção dos nós e pesos[1, 2, 43]:

$$\sin^{4i}(\phi), \sin^{2j+1}(\phi), \sin^{4k+1}(\phi)\cos(\phi), \tag{4.29}$$

com i, j e k inteiros não negativos, satisfazendo $i < N_\phi/4$, $j < (N_\phi - 1)/2$ e $k < (N_\phi - 2)/4$. Por exemplo, para $N_\phi = 2$, é suficiente considerar os integrandos 1 e $\sin(\phi)$, e devido as condições de simetria, resolvemos o seguinte sistema não linear:

$$2w_1^a = \int_0^{\pi/2} 1 d\phi \tag{4.30}$$

$$w_1^a \sin(\phi_1) + w_1^a \cos(\phi_1) = \int_0^{\pi/2} \sin(\phi) d\phi. \tag{4.31}$$

Na próxima seção, discutiremos os aspectos computacionais para a geração da quadratura QR via sistemas não lineares.

4.2.2 Aspectos computacionais da quadratura QR via sistemas não lineares

Neste estudo, buscamos entender como são gerados os nós e pesos das quadraturas azimutal e polar discutidas na seção anterior, replicar os resultados

encontrados em Abu-Shumays [2] e por fim, elaborar um algoritmo para geração e resolução destes sistemas para ordens N_θ e N_ϕ arbitrárias. Para tal, utilizamos o software Scilab 6.0.2 [27], juntamente com o pacote NUMERICO de Da Cunha [19].

Iniciamos este estudo pela variável azimutal, pois para esta variável Abu-Shumays [2] gerou numericamente conjuntos de quadratura de ordens mais altas que para a variável polar. Conforme discutido na seção anterior, geramos o sistema não linear associado a variável azimutal utilizando os integrandos dados pela equação (4.29), e realizamos a seguinte mudança de variável:

$$\sin(\phi) = x \rightarrow \cos(\phi) = \sqrt{1 - x^2}. \tag{4.32}$$

Com essa mudança de variável, o sistema não linear passa a ser calculado para a variável x e não para a variável ϕ. Por fim, para o cálculo do termo independente do sistema não linear, fazemos uso dos seguintes resultados [2]:

$$\int_0^{\pi/2} \sin^{4i}(\phi) d\phi = \frac{1.3.5 \cdots (4i-1)}{2^{2i+1}(2i)!}, \tag{4.33}$$

$$\int_0^{\pi/2} \sin^{2j+1}(\phi) d\phi = \frac{2^j j!}{1.3.5 \cdots (2j+1)}, \tag{4.34}$$

$$\int_0^{\pi/2} \sin^{4k+1}(\phi) \cos(\phi) d\phi = \frac{1}{4k+2}. \tag{4.35}$$

Uma vez que utilizamos as condições de simetria que os nós e pesos devem satisfazer, não é necessário resolver um sistema não linear para o caso $N_\phi = 1$. Para este caso, só é necessário computar o valor do peso, w_1^a, uma vez que $\phi_1 = \pi/4$. Abaixo, exemplificamos os sistemas gerados para $N_\phi = 2, \ldots, 5$.

- $N_\phi = 2$:

$$\begin{cases} 2w_1^a = \pi/2 \\ w_1^a x_1 + w_1^a \sqrt{1 - x_1^2} = 1. \end{cases} \tag{4.36}$$

- $N_\phi = 3$:

$$\begin{cases} 2w_1^a + w_2^a = \pi/2 \\ w_1^a x_1 + w_1^a \sqrt{1 - x_1^2} + \frac{\sqrt{2}}{2} w_2^a = 1 \\ 2w_1^a x_1 \sqrt{1 - x_1^2} + \frac{1}{2} w_2^a = 1/2. \end{cases} \tag{4.37}$$

- $N_\phi = 4$:

$$\begin{cases} 2w_1^a + 2w_2^a = \pi/2 \\ w_1^a x_1 + w_1^a \sqrt{1 - x_1^2} + w_2^a x_2 + w_2^a \sqrt{1 - x_2^2} = 1 \\ 2w_1^a x_1 \sqrt{1 - x_1^2} + 2w_2^a x_2 \sqrt{1 - x_2^2} = 1/2 \\ w_1^a x_1^3 + w_1^a (\sqrt{1 - x_1^2})^3 + w_2^a x_2^3 + w_2^a (\sqrt{1 - x_2^2})^3 = 2/3. \end{cases} \tag{4.38}$$

- $N_\phi = 5$:

$$\begin{cases} 2w_1^a + 2w_2^a + w_3^a = \pi/2 \\ w_1^a x_1 + w_1^a \sqrt{1 - x_1^2} + w_2^a x_2 + w_2^a \sqrt{1 - x_2^2} + \frac{\sqrt{2}}{2} w_3^a = 1 \\ 2w_1^a x_1 \sqrt{1 - x_1^2} + 2w_2^a x_2 \sqrt{1 - x_2^2} + \frac{1}{2} w_3^a = 1/2 \\ w_1^a x_1^3 + w_1^a (\sqrt{1 - x_1^2})^3 + w_2^a x_2^3 + w_2^a (\sqrt{1 - x_2^2})^3 + (\frac{\sqrt{2}}{2})^3 w_3^a = 2/3 \\ w_1^a x_1^4 + w_1 (\sqrt{1 - x_1^2})^4 + w_2^a x_2^4 + w_2 (\sqrt{1 - x_2^2})^4 + \frac{1}{4} w_3^a = 3\pi/16. \end{cases} \tag{4.39}$$

Desta forma, elaboramos um código computacional para gerar os sistemas não lineares para uma ordem N_ϕ arbitrária. Para a resolução destes sistemas, utilizamos a rotina *num_ newton* do pacote NUMERICO. Tal rotina é a implementação do método de Newton para a resolução de sistemas de equações não lineares [46], que calcula uma aproximação para a solução do sistema não linear $F(x) = 0$, com estimativa inicial o vetor x_0, sujeita a

$$||F(x)|| < \tau_r||F(x_0)|| + \tau_a, \tag{4.40}$$

desde que não exceda a $kmax$ iterações, com

- $F(x)$: descrição do sistema não linear; vetor coluna contendo as funções expressas como "strings";
- x_0: vetor coluna; estimativa inicial da solução;
- τ_r: valor real; fator multiplicativo da norma L-2 de $F(x)$ avaliada na estimativa inicial;
- τ_a: valor real; fator aditivo na tolerância ($\tau_a > \tau_r$);
- $kmax$: valor inteiro; número máximo de iterações.

A seguir, apresentamos três casos testes com diferentes aproximações iniciais.

4.2.2.1 Caso 1

Neste caso, utilizamos como aproximação inicial o vetor nulo, e como tolerâncias, $\tau_r = 1E-16$, $\tau_a = 1E-14$ e $kmax = 1000$. Com estes parâmetros e

aproximação inicial, obtemos convergência para as ordens $N_\phi = 1, 2, \ldots, 5$. Para as ordens $N_\phi = 6, 7, \ldots, 22$, o sistema não convergiu. Na tabela 4.3 listamos o número de iterações necessárias para a convergência utilizando como aproximação inicial o vetor nulo.

Tabela 4.3 Ordens convergentes e respectivo número de iterações para o caso 1

N_ϕ	1	2	3	4	5
k	2	6	6	8	9

Nota: k representa o número de iterações necessárias para a convergência

Como conseguimos replicar uma quantidade pequena de ordens (Abu-Shumays [2] gerou resultados para $N_\phi \leq 22$), buscamos outra alternativa para a aproximação inicial, visto que a convergência do método de Newton está intimamente associada a ela [46].

4.2.2.2 Caso 2

Para este caso, mantivemos as mesmas tolerâncias utilizadas no caso 1, mas utilizamos como aproximação inicial um vetor contendo 1 em todas as entradas. Com estes parâmetros e aproximação inicial, obtivemos convergência para as ordens $N_\phi = 1, 2, \ldots, 6$, enquanto que as demais ordens continuaram divergindo. Com esta nova aproximação inicial, notamos que o número de iterações necessárias para a convergência das ordens $N_\phi = 1, \ldots, 5$ aumentou (como pode ser visto na tabela 4.4), principalmente para a ordem $N_\phi = 5$, em que o aumento foi de 140 iterações.

Tabela 4.4 Ordens convergentes e respectivo número de iterações para o caso 2

N_ϕ	1	2	3	4	5	6
k	2	8	9	17	149	132

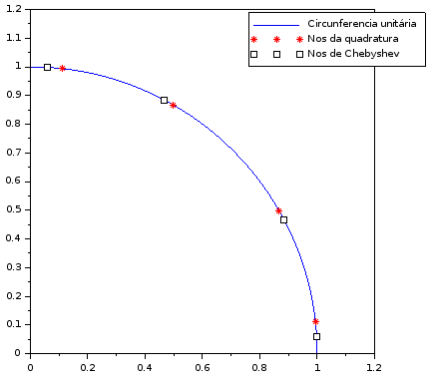
Nota: k representa o número de iterações necessárias para a convergência

Assim como ocorreu com o primeiro caso, no segundo caso conseguimos replicar os resultados para baixas ordens, sendo que a mudança feita em nossa solução inicial não foi eficaz. Dessa forma, no terceiro caso refinamos nossa aproximação inicial e esperamos que isso resulte em melhores resultados.

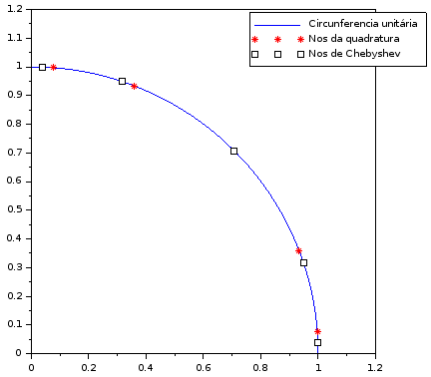
4.2.2.3 Caso 3

No terceiro caso, mantivemos os mesmos parâmetros para a resolução dos sistemas não lineares utilizados anteriormente, e procuramos uma aproximação inicial mais próxima da solução do sistema não linear. Salientamos aqui que a aproximação inicial é composta por duas partes: uma parte é a aproximação inicial para os nós e a outra parte é a aproximação inicial para os pesos.

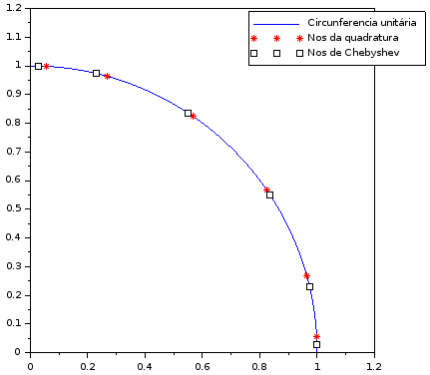
Como forma de encontrar uma melhor aproximação inicial, procuramos aproximar os nós da quadratura pelos nós de Chebyshev mapeados para o intervalo $[0, \pi/2]$. Como pode ser visto na figura 4.5, esta escolha de aproximação inicial para os nós é uma boa aproximação para os nós da quadratura.



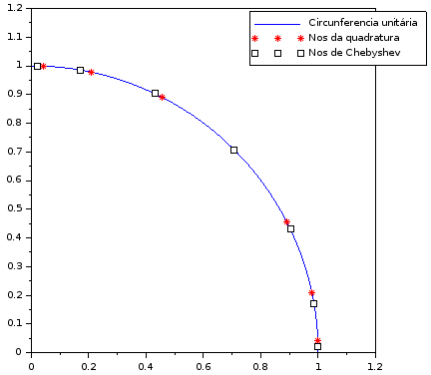
(a) $N_\phi = 4$



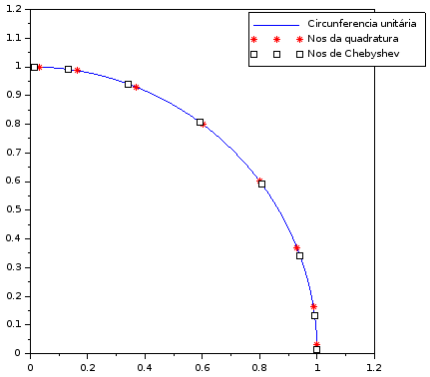
(b) $N_\phi = 5$



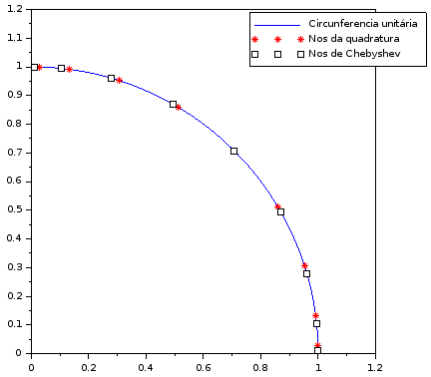
(c) $N_\phi = 6$



(d) $N_\phi = 7$



(e) $N_\phi = 8$



(f) $N_\phi = 9$

Figura 4.5 Aproximação inicial para os nós

Para encontrarmos uma boa aproximação inicial relacionada aos pesos, notamos que, em relação a eles, temos um sistema linear. Com isso, utilizando a aproximação para os nós descritas acima, resolvemos um sistema linear através de mínimos quadrados e obtemos uma aproximação inicial para os pesos.

Utilizando esta aproximação inicial e o método de Newton com os parâmetros anteriores, obtemos convergência somente para as ordens $N_\phi = 1, 2, \ldots, 7$, e para as demais ordens o sistema divergiu. Na tabela 4.5 listamos o número de iterações necessárias para a convergência utilizando esta aproximação inicial.

Tabela 4.5 Ordens convergentes e respectivo número de iterações para o caso 3

N_ϕ	1	2	3	4	5	6	7
k	4	4	5	5	5	5	7

Nota: k representa o número de iterações necessárias para a convergência

Elaboramos um código computacional para gerar a aproximação inicial com uma ordem N_ϕ arbitrária. Contudo, como obtivemos convergência somente para $N_\phi \leq 7$, verificamos o número de condição da matriz Jacobiana do sistema não linear utilizada no método de Newton, bem como a norma da inversa da matriz Jacobiana, para $N_\phi = 8, 9, \ldots, 22$. As tabelas 4.6 e 4.7 listam os valores obtidos para a Jacobiana calculada em um vetor aleatório e na solução dada em Abu-Shumays [2], respectivamente. Através dos valores obtidos, confirmamos o mal-condicionamento do sistema não linear já mencionado por Abu-Shumays [2]. Ainda, segundo Abu-Shumays [2], este mal-condicionamento do sistema se deve ao fato de que os termos envolvidos no sistema de equações não lineares combinam várias potências de x_i e $\sqrt{1 - x_i}$, com alguns valores x_i próximos de zero e outros próximos de 1.

Tabela 4.6 Condicionamento da Jacobiana e norma da Inversa da Jacobiana aplicada em vetores aleatórios

N_ϕ	**Condicionamento da matriz Jacobiana**	**Norma da inversa da matriz Jacobiana**
8	9.225D+11	1.018D+11
9	1.758D+12	5.510D+10
10	4.544D+12	4.721D+10
11	1.038D+12	1.643D+12
12	2.241D+12	1.169D+11
13	2.634D+12	4.769D+11
14	1.270D+14	3.082D+11
15	2.639D+12	3.740D+11
16	3.075D+12	4.077D+11
17	1.468D+13	1.809D+11
18	3.051D+12	1.053D+13
19	1.357D+12	1.399D+12
20	1.537D+13	1.157D+11
21	2.108D+12	7.295D+11
22	7.048D+12	2.761D+11

Tabela 4.7 Condicionamento da Jacobiana e norma da Inversa da Jacobiana aplicada na solução tabelada em Abu-Shumays[2]

N_ϕ	**Condicionamento da matriz Jacobiana**	**Norma da inversa da matriz Jacobiana**
8	1.173D+10	2.036D+09
9	2.043D+09	3.325D+08
10	1.827D+09	2.692D+08
11	5.267D+09	7.516D+08
12	2.530D+09	3.325D+08
13	1.317D+10	1.648D+09
14	2.545D+10	2.975D+09
15	5.911D+09	6.735D+08
16	4.480D+10	4.811D+09
17	8.236D+10	8.494D+09
18	4.203D+11	4.116D+10
19	5.518D+10	5.289D+09
20	6.635D+10	6.070D+09
21	1.856D+11	1.634D+10
22	5.644D+11	4.788D+10

Também testamos os valores $\tau_r = 1E-13$ e $\tau_a = 1E-12$, com a aproximação inicial descrita acima, mas isto não afetou os resultados, de modo que a convergência continuou sendo alcançada somente para $N_\phi \leq 7$. Durante a execução do método de Newton para resolver o sistema não linear, observamos que as primeiras iterações aparentavam estar convergindo, mas a partir de um determinado momento passavam a divergir. Devido a esse comportamento, modificamos o passo da iteração do método de Newton, e testamos os valores $h/2$, $h/4$, $h/8$, $h/16$ e $h/32$, com τ_r, τ_a e kmax iguais a $1E-16$, $1E-14$ e 1000, respectivamente, em que h representa

o passo de iteração calculado no método de Newton. Utilizando estes parâmetros e o passo modificado, obtivemos convergência para $N_\phi \leq 8$ para os valores $h/2$, $h/4$, $h/16$, sendo que para as ordens $N_\phi > 8$ não houve convergência para nenhum valor de h testado.

Mantendo a mesma aproximação inicial descrita acima, investigamos o método de Armijo [46] para resolver o sistema não linear, mas não demonstrou bom desempenho, sendo que obtivemos convergência somente para as mesmas ordens que o método de Newton.

Até o momento, procuramos soluções para o sistema não linear na variável x, em que fizemos a mudança de variável dada pela equação (4.32). Como já temos um código computacional gerando os sistemas não lineares e a aproximação inicial para uma ordem N_ϕ arbitrária, modificamos este código e investigamos o uso da variável ϕ, isto é, escrevemos as equações do sistema não linear em termos de *senos* e *cossenos*. Contudo, os resultados obtidos foram análogos aos anteriores, não obtendo convergência para $N_\phi \geq 8$. Em nossa última tentativa de resolver o sistema não linear associado a variável azimutal, escrevemos o sistema não linear de forma que, ao invés de considerarmos somente os integrandos dados pela equação (4.29), utilizamos todos os integrandos da forma

$$\sin^l(\phi)\cos^m(\phi), \tag{4.41}$$

com $l+m \leq N_\phi - 1$. Deste modo, obtemos um sistema de equações não lineares com mais equações que incógnitas, mas os resultados obtidos são análogos aos anteriores.

Em relação à variável polar, Abu-Shumays [1, 2] diz que este sistema também é mal-condicionado, e devido a isso, foram gerados numericamente somente os nós e pesos de quadratura para $N_\phi \leq 7$ [1], e posteriormente estendidos para

$N_\phi \leq 10$ [2]. De modo análogo ao tratamento da variável azimutal, implementamos um código computacional para geração do sistema não linear, e resolvemos através da rotina *num_newton* do pacote NUMERICO, com parâmetros $\tau_r = 1E-16$, $\tau_a = 1E-14$ e $kmax = 1000$, utilizando como condição inicial um vetor nulo e um vetor com todas as entradas iguais a um. Em ambos casos, conseguimos convergência somente para a ordem $N_\theta = 1$.

Devido as dificuldades encontradas na resolução dos sistemas não lineares, tanto para a variável azimutal quanto para a variável polar, Spence [70] propôs uma abordagem fazendo uso de polinômios ortogonais, discutidos no capítulo 3. Na seção seguinte, apresentamos detalhes desta abordagem.

4.2.3 Quadratura QR via polinômios ortogonais

Para contornar as dificuldades encontradas na resolução dos sistemas não lineares na obtenção das quadraturas unidimensionais associadas as variáveis polar e azimutal, Spence [70] propôs uma abordagem na qual determinamos os nós e pesos destas quadraturas unidimensionais a partir de polinômios ortogonais não clássicos. De forma mais clara, o problema de obtenção de nós e pesos de um esquema de quadratura é transformado no cálculo de autovalores de uma matriz tridiagonal simétrica gerada a partir das fórmulas de recorrência de polinômios ortogonais, conforme visto no capítulo 3.

Conforme discutido na seção 3.1, os polinômios ortogonais satisfazem a seguinte relação de recorrência de três termos:

$$\begin{cases} \pi_{k+1}(x) & = (x-\alpha_k)\pi_k(x) - \beta_k\pi_{k-1}(x), \text{ para } k = 1, 2, \ldots \\ & \pi_0(x) = 1 \ , \ \pi_1(x) = x - \frac{\langle x,1\rangle}{\langle 1,1\rangle}, \end{cases} \tag{4.42}$$

em que

$$\alpha_k = \frac{\langle x\pi_k(x), \pi_k(x)\rangle}{\langle \pi_k(x), \pi_k(x)\rangle} \ , \ \beta_k = \frac{\langle \pi_k(x), \pi_k(x)\rangle}{\langle \pi_{k-1}(x), \pi_{k-1}(x)\rangle} \ , \text{ para } k = 1, 2, \ldots, \tag{4.43}$$

e

$$\langle f(x), g(x)\rangle = \int_a^b \omega(x) f(x) g(x) dx. \tag{4.44}$$

Por meio de manipulações algébricas na relação de recorrência (4.42), mostramos (na seção 3.2) que as raízes de $\pi_n(x)$ são determinadas como os autovalores de

$$J_n = \begin{pmatrix} \alpha_1 & \sqrt{\beta_1} & 0 & 0 & \cdots & 0 \\ \sqrt{\beta_1} & \alpha_2 & \sqrt{\beta_2} & 0 & \cdots & 0 \\ 0 & \sqrt{\beta_2} & \alpha_3 & \sqrt{\beta_3} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \sqrt{\beta_{n-2}} & \alpha_{n-1} & \sqrt{\beta_{n-1}} \\ 0 & \cdots & \cdots & 0 & \sqrt{\beta_{n-1}} & \alpha_n \end{pmatrix}. \tag{4.45}$$

Contudo, uma dificudade relevante neste caso é o cálculo das constantes α_k e β_k, que são as entradas da matriz J_n, pois envolve a avaliação de integrais cuja avaliação não podem ser dadas por fórmulas fechadas. Para contornar isso, Spence propôs avaliar numericamente tais integrais através de uma quadratura numérica adequada que discutiremos mais tarde, e sumarizou os passos necessários para a obtenção dos nós e pesos de quadratura, que são apresentados a seguir:

1. Defna $\pi_0(x) = 1$;

2. Para $j = 0$ até $n - 1$ faça:

   2.1 Usando um esquema de quadratura, determine os produtos internos

   $$\langle x\pi_j(x), \pi_j(x)\rangle = \int_a^b x\pi_j^2(x)\omega(x)dx, \tag{4.46}$$

   e

   $$\langle \pi_j(x), \pi_j(x)\rangle = \int_a^b \pi_j^2(x)\omega(x)dx; \tag{4.47}$$

   2.2 Calcule as constantes α_j e β_j, utilizando as integrais calculadas acima;

   2.3 Compute o polinômio ortogonal de grau mais alto utilizando a relação de recorrência (4.42);

3. Uma vez obtidas todas as constantes α_k e β_k, preencha a matriz J_n;

4. Utilizando um método numérico adequado, determine os autovalores e autovetores normalizados J_n;

5. Determine os pesos de quadratura, conforme discutido na seção 3.2.

Nas seções seguintes, discutimos como são geradas as quadraturas para a variável polar e para a variável azimutal através desta abordagem.

4.2.3.1 Quadratura associada à variável polar

Para a quadratura associada a variável polar, Spence [70] escreve a integral sobre o ângulo polar dada pela equação (2.35) da mesma forma que Abu-Shumays

[1], como visto na seção (4.2.1), isto é,

$$\int_0^{\pi} \sin^{l+m}\theta \sin\theta d\theta = 2\int_0^{\pi/2} \sin^{l+m}\theta \sin\theta d\theta \tag{4.48}$$

$$= 2\int_0^1 t^k \frac{tdt}{\sqrt{1-t^2}} \tag{4.49}$$

em que foram feitas as mudanças de variáveis $\sin(\theta) = t$, e $k = l + m$.

Contudo, Spence utiliza a abordagem via polinômios ortogonais. Dessa forma, os nós da quadratura associada a variável polar de ordem N_θ são as raízes do polinômio ortogonal $\pi_{N_\theta}(t)$ associado a função peso

$$\omega_1(t) = \frac{t}{\sqrt{1-t^2}}, \tag{4.50}$$

no intervalo $[0, 1]$. Uma vez que a função peso é conhecida e não negativa, os polinômios ortogonais associados a ω_1 no intervalo $[0, 1]$ podem ser gerados e, consequentemente, a correspondente quadratura gaussiana pode ser extraída com os passos sumarizados anteriormente.

Como podemos ver, a função peso ω_1 possui uma singularidade em $t = 1$. Para contornar isto na avaliação dos produtos internos, Spence [70] propôs escrever tais produtos internos como

$$\langle t\pi_i(t), \pi_i(t)\rangle = \int_0^1 t(u)\pi_i^2(t(u))du \tag{4.51}$$

e

$$\langle \pi_i(t), \pi_i(t)\rangle = \int_0^1 \pi_i^2(t(u))du, \tag{4.52}$$

em que $t(u) = \sqrt{1-u^2}$, e em termos das equações (4.51) e (4.52), a função peso é 1, de modo que agora não há singularidade. Uma vez gerado o conjunto de abscissas,

$\{t_i\}$, devido a mudança de variável feita na equação (4.49), podemos reescrevê-lo em termos do ângulo polar, θ, notando que

$$\theta_i = \arcsin(t_i). \tag{4.53}$$

Por fim, os pesos desta quadratura devem somar 1.

A seguir, apresentamos cinco conjuntos de quadratura para a variável azimutal. Tais conjuntos se diferem pelas condições de simetria exigida dos nós e pelo conjunto de polinômios ortogonais utilizados na sua geração.

4.2.3.2 Quadratura associada à variável azimutal

Para a quadratura associada à variável azimutal, Spence [70] tenta determinar os pesos, w_j^a, e abscissas, ϕ_j, que representam a regra

$$\int_0^{\pi/2} \cos^l(\phi) \sin^m(\phi) d\phi \approx \sum_{j=1}^{N_\phi} w_j^a \cos^l(\phi_j) \sin^m(\phi_j), \tag{4.54}$$

(ou variações disto) de forma exata para diferentes valores de l e m. De acordo com Spence [70], vários conjuntos de quadratura podem ser obtidos a partir da equação (4.54), dependendo dos requerimentos de simetria para os nós e de restrições para os valores de l e m.

A seguir, apresentamos três conjuntos de quadratura para a variável azimutal que possuem simetria em relação a $\phi = \pi/4$.

Azimutal-QRS45: Esta é uma quadratura proposta por Spence [70] com simetria em relação a $\phi = \pi/4$. Para gerá-la, Spence procura uma quadratura

para aproximar a integral

$$\int_0^{\pi/2} \cos^{2v}(\phi - \frac{\pi}{4}) \sin^m(\phi - \frac{\pi}{4}) d\phi = \int_{-1/\sqrt{2}}^{1/\sqrt{2}} (1-t^2)^v t^m \omega_2(t) dt$$
$$\approx \sum_{j=1}^{N_\phi} w_j^a (1-t_j^2)^v t_j^m, \tag{4.55}$$

em que é feita a restrição de que l deve ser par, isto é, $l = 2v$ (v sendo um inteiro), e a substituição $t = \sin(\phi - \frac{\pi}{4})$. A função peso é dada por

$$\omega_2(t) = \frac{1}{\sqrt{1-t^2}}, \tag{4.56}$$

que é estritamente positiva no intervalo de integração.

Uma vez que a função peso, ω_2, é conhecida, podemos gerar o conjunto de abscissas, $\{t_j\}$, e reescrevê-lo em termos do ângulo azimutal, ϕ, notando que

$$\phi_j = \arcsin(t_j) + \pi/4. \tag{4.57}$$

Por fim, a soma dos pesos de quadratura deve ser $\pi/2$.

Azimutal-QRA45: Esta também é proposta por Spence [70], e reproduz a quadratura proposta por Abu-Shumays [2] discutida na seção 4.2.1. Para gerá-la, Spence procura uma quadratura para aproximar a integral

$$\int_0^{\pi/2} \cos^{2v}\left[\frac{1}{2}(\phi - \frac{\pi}{4})\right] \sin^m\left[\frac{1}{2}(\phi - \frac{\pi}{4})\right] d\phi = \int_{-b}^{b} (1-t^2)^v t^m \omega_3(t) dt$$
$$\approx \sum_{j=1}^{N_\phi} w_j^a (1-t_j^2)^v t_j^m, \tag{4.58}$$

em que é feita a restrição de que l deve ser par e a substituição $t = \sin[\frac{1}{2}(\phi - \frac{\pi}{4})]$. Os limites de integração são

$$\pm b = \pm \sin(\frac{\pi}{8}), \tag{4.59}$$

e a função peso dada por

$$\omega_3(t) = \frac{2}{\sqrt{1-t^2}}, \tag{4.60}$$

que é positiva sobre o domínio de integração, (4.59).

Uma vez gerado o conjunto de abscissas, $\{t_j\}$, podemos reescrevê-lo em termos do ângulo azimutal, ϕ, notando que

$$\phi_j = 2\arcsin(t_j) + \pi/4. \tag{4.61}$$

Por fim, a soma dos pesos de quadratura deve ser $\pi/2$.

Azimutal-QRJ45: Este é um esquema de quadratura com simetria em relação a $\phi = \pi/4$ que propomos. Para tal, procuramos uma quadratura para aproximar a integral

$$\begin{aligned}\int_0^{\pi/2} \cos^{2v}\left[2(\phi - \frac{\pi}{4})\right] \sin^m\left[2(\phi - \frac{\pi}{4})\right] d\phi &= \frac{1}{2}\int_{-1}^{1} (1-t^2)^v t^m \omega_2(t) dt \\ &\approx \sum_{j=1}^{N_\phi} w_j^a (1-t_j^2)^v t_j^m, \end{aligned} \tag{4.62}$$

em que fizemos a restrição de que l deve ser par e a substituição $t = \sin\left[2(\phi - \frac{\pi}{4})\right]$. A função peso, $\omega_2(t)$, é a mesma apresentada na equação (4.56), e é estritamente positiva no intervalo de integração.

Note que, devido a função peso, $\omega_2(x)$, o intervalo de integração e a menos do fator 1/2, procuramos nós e pesos para a quadratura de Gauss-Chebyshev, que já foi discutida na seção 3.4. Portanto,

$$t_j = \cos\left(\frac{(2j-1)}{2N_\phi}\pi\right) \tag{4.63}$$

e

$$w_j^a = \frac{\pi}{2N_\phi} \tag{4.64}$$

para $j = 1, 2, \ldots, N_\phi$. Por fim, podemos reescrever o conjunto de abscissas, $\{t_j\}$, em relação ao ângulo azimutal, ϕ, notando que

$$\phi_j = \frac{\pi}{2} - \frac{(2j-1)}{4N_\phi}\pi, \tag{4.65}$$

e a soma dos pesos desta quadratura deve ser $\pi/2$.

Os próximos dois esquemas de quadraturas não possuem simetria, sendo um deles proposto por Spence [70] e outro que nós estamos propondo.

Azimutal-QRS90: Esta é uma quadratura proposta por Spence [70] sem simetria. Para gerá-la, Spence procura uma quadratura para aproximar a integral

$$\int_0^{\pi/2} \cos^m(\phi)\sin^m(\phi)d\phi \approx \sum_{j=1}^{N_\phi} w_j^a \cos^m(\phi_j)\sin^m(\phi_j), \tag{4.66}$$

em que é feita a restrição de que $l = m$. Se fizermos a substituição $t = \frac{1}{2}\sin(2\phi)$, a integral do lado esquerdo da equação (4.66) deve ser dividida em duas e posteriormente recombinadas, de modo que obtemos

$$\int_0^{\pi/2} \left(\frac{1}{2}\sin(2\phi)\right)^m d\phi = \int_0^{1/2} t^m \omega_4(t)dt, \tag{4.67}$$

em que a função peso é dada por

$$\omega_4(t) = \frac{2}{\sqrt{1-4t^2}}, \tag{4.68}$$

que é positiva sobre o domínio de integração.

Note que, a função peso $\omega_4(t)$ possui uma singularidade em $t = 1/2$, e para contorná-la na avaliação dos produtos internos, Spence [70] propôs escrever

estes produtos internos como

$$\langle t\pi_j(t), \pi_j(t)\rangle = \int_0^{\pi/2} t(u)\pi_j^2(t(u))du \tag{4.69}$$

e

$$\langle \pi_j(t), \pi_j(t)\rangle = \int_0^{\pi/2} \pi_i^2(t(u))du, \tag{4.70}$$

em que $t(u) = \frac{1}{2}\sin(u)$, e em termos das equações (4.69) e (4.70), a função peso é 1, de modo que agora não há singularidade. Uma vez gerado o conjunto de abscissas, $\{t_j\}$, podemos reescrevê-lo em termos do ângulo azimutal, ϕ, notando que

$$\phi_j = \arcsin(2t_j), \tag{4.71}$$

e a soma dos pesos de quadratura deve ser $\pi/2$.

Azimutal-QRJ90: Por fim, propomos este esquema de quadratura sem simetria. Para tal, procuramos uma quadratura para aproximar a integral

$$\begin{aligned}\int_0^{\pi/2} \cos^{2v}(\frac{\phi}{2})\sin^m(\frac{\phi}{2})d\phi &= \int_0^{1/\sqrt{2}} (1-t^2)^v t^m \omega_3(t)dt \\ &\approx \sum_{j=1}^{N_\phi} w_j^a (1-t_j^2)^v t_j^m,\end{aligned} \tag{4.72}$$

em que fizemos a restrição de que l deve ser par e a substituição $t = \sin(\frac{\phi}{2})$. A função peso, $\omega_3(t)$, é a mesma apresentada na equação (4.60), e é estritamente positiva no intervalo de integração.

Uma vez gerado o conjunto de abscissas, $\{t_j\}$, podemos reescrevê-lo em termos do ângulo azimutal, ϕ, notando que

$$\phi_j = 2\arcsin(t_j), \tag{4.73}$$

e a soma dos pesos de quadratura deve ser $\pi/2$.

Na seção seguinte, apresentamos os aspectos computacionais associados da quadratura QR via polinômios ortogonais.

4.2.4 Aspectos computacionais da quadratura QR via polinômios ortogonais

Como dito anteriormente, uma dificuldade relevante encontrada nesta abordagem está na obtenção de α_k e β_k, que são as entradas da matriz J_n, pois envolve a avaliação de integrais que não podem ser representadas em fórmulas fechadas. Para contornar isso, Spence [70] propôs avaliar numericamente tais integrais através do esquema de quadratura *função erro* [6, 13].

O esquema de quadratura *função erro* é baseado na fórmula de somatório de Euler-Maclaurin, que pode ser definida da seguinte forma [4]: Sejam os inteiros $m \geq 0$ e $n \geq 1$, e defina $h = (b-a)/n$ e $x_j = a + jh$ para $0 \leq j \leq n$. Assuma também que a função $f(x)$ possui pelo menos derivada de ordem $2m+2$ em $[a, b]$. Então,

$$\begin{aligned}\int_a^b f(x)dx =& h\sum_{j=0}^{n} f(x_j) - \frac{h}{2}(f(a)+f(b)) \\ &- \sum_{i=1}^{m} \frac{h^{2i}B_{2i}}{(2i)!}\left(f^{(2i-1)}(b) - f^{(2i-1)}(a)\right) - E,\end{aligned} \tag{4.74}$$

em que B_{2i} denota os números de Bernoulli, e

$$E = \frac{h^{2m+2}(b-a)B_{2m+2}f^{(2m+2)}(\chi)}{(2m+2)!}, \tag{4.75}$$

para algum $\chi \in (a, b)$.

Na circunstância em que a função $f(x)$ e todas as suas derivadas são zero nos pontos de extremidade a e b, note que o segundo e o terceiro termos da fórmula

de Euler-Maclaurin são zero. Para tais funções, o erro de uma simples aproximação da integral por uma função passo, com intervalo h, é simplesmente E. Mas como E é menor que uma constante multiplicada por $h^{2m+2}/(2m+2)!$, para qualquer m, concluímos que o erro tende a zero mais rapidamente que qualquer potência de h. No caso de um função definida em $(-\infty, \infty)$, a fórmula de Euler-Maclaurin ainda se aplica à aproximação resultante, desde que a função e todas as suas derivadas tendam a zero para grandes argumentos positivos e negativos [6, 13].

O princípio utilizado no esquema de quadratura *função erro* é transformar a integral de $f(x)$ sobre um intervalo finito (que tomaremos sobre $(-1, 1)$, por conveniência), para uma integral sobre $(-\infty, \infty)$ usando a mudança de variável $x = g(t)$ [13]. Aqui, $g(x)$ é uma função monotônica com a propriedade que $g(x) \to 1$ quando $x \to \infty$, e $g(x) \to -1$ quando $x \to -\infty$, e também com a propriedade de que $g'(x)$ e todas as suas derivadas de ordem superior rapidamente se aproximam de zero para argumentos grandes [6, 13]. Neste caso podemos escrever [13], para $h > 0$,

$$\int_{-1}^{1} f(x)dx = \int_{-\infty}^{\infty} f(g(t))g'(t)dt = h\sum_{-\infty}^{\infty} w_j f(x_j), \tag{4.76}$$

em que $x_j = g(jh)$ e $w_j = g'(jh)$. No caso da quadratura *função erro*, temos que $g(t) = erf(t)$ e $g'(t) = (2/\sqrt{\pi})e^{-t^2}$ [6, 13]. Note que, a quadratura *função erro* aproxima a integral através de uma soma infinita. Dessa forma, truncamos a soma infinita da equação (4.76) quando $w_j < tol\_erf$, em que tol_erf é uma tolerância que nós determinamos. Assim, para a implementação desta quadratura, são necessários a determinação de dois parâmetros: h e tol_erf, responsáveis pelo passo de quadratura e critério de parada, respectivamente. Caso o intervalo de integração (a, b) seja outro que não $(-1, 1)$, basta fazermos um mapeamento do intervalo (a, b) para o intervalo $(-1, 1)$ [13].

Com isso, Spence apresentou um algoritmo no software Matlab [72] para a geração destas quadraturas. Em seu algoritmo, o problema de autovalores é realizado pela função *eig* (que é um intrínseco do Matlab), os pesos da quadratura são calculados a partir da equação (3.44) e as constantes α_k e β_k são calculadas a partir do esquema de quadratura *função erro*. O algoritmo implementado por Spence pode ser encontrado em [70].

Motivados por este algoritmo, procuramos implamentá-lo em linguagem Fortran 95 em precisão simples, dupla e quádrupla, seguindo três etapas. Na primeira etapa, tratamos da implementação somente em precisão dupla e a mais próxima possível do algoritmo apresentado por Spence em [70]. Sendo assim, não fizemos nenhum tipo de otimização no algoritmo, e as únicas mudanças feitas foram as necessárias para escrita em Fortran 95, tais como inicialização de variáveis e alocação de vetores. Para a resolução do problema de autovalores e autovetores, utilizamos a rotina *DSTEVD* da biblioteca LAPACK [3]. No decorrer deste trabalho, iremos nos referir a nossa implementação em linguagem Fortran 95 feita na primeira etapa como *fn_qrs_full_erf_spence*. Uma dificuldade inicial que surge na implementação do algoritmo *fn_qrs_full_erf_spence* está relacionada a função *erf*, pois ela não é uma função intrínseca da linguagem Fortran 95. Já que estamos utilizando o compilador gfortran 7.4.0, contornamos esta dificuldade utilizando a função *erf* deste compilador.

No algoritmo *fn_qrs_full_erf_spence*, bem como no algoritmo proposto por Spence, cada quadratura é identificada por uma variável inteira chamada *flag*. A partir de cada valor da variável *flag*, o algoritmo realiza a inicialização dos parâmetros necessários para a obtenção da quadratura associada a esta *flag*, tais como o intervalo de integração, $[a, b]$, a função peso utilizada, e gera os nós e pesos de quadratura da

função erro, utilizados no cálculo das constantes α_k e β_k. A tabela 4.8 apresenta a descrição de cada *flag* utilizada neste trabalho.

Tabela 4.8 Valores da variável *flag* e respectiva quadratura

Flag	**Quadratura**
1	Azimutal-QRS45
2	Azimutal-QRA45
3	Azimutal-QRS90
4	Polar
5	Azimutal-QRJ45
6	Azimutal-QRJ90

Para verificarmos o tempo computacional do algoritmo, transcrevemos o algoritmo proposto por Spence no software Matlab R2018b [72] e calculamos o tempo médio de 1000 execuções para obtenção destas quadraturas unidimensionais de ordens $N = 10, 20, \ldots, 100$. Realizamos o mesmo procedimento para o algoritmo *fn_qrs_full_erf_spence*, utilizando o mesmo computador descrito na seção 4.1.3, bem como o mesmo compilador para Fortran 95 e rotina utilizada para obtenção dos tempos. Os valores obtidos estão nas tabelas 4.9 e 4.10, em que utilizamos os seguintes parâmetros para a quadratura *função erro*: $h = 0.004$ e $tol\_erf = 1E-17$. Tais parâmetros são os mesmos apresentados por Spence [70].

Tabela 4.9 Tempo médio de execução (em segundos) do algoritmo apresentado por Spence e implementado no Matlab R2018b

Ordem	**Flag = 1**	**Flag = 2**	**Flag = 3**	**Flag = 4**	**Flag = 5**	**Flag = 6**
10	1,1846E-02	1,0924E-02	1,0444E-02	1,0283E-02	1,8991E-05	1,0894E-02
20	1,4667E-02	1,2395E-02	1,0859E-02	1,0922E-02	1,7978E-05	1,2335E-02
30	1,4757E-02	1,4120E-02	1,1897E-02	1,1849E-02	1,5677E-05	1,4139E-02
40	1,6218E-02	1,5654E-02	1,2847E-02	1,2801E-02	1,5463E-05	1,5464E-02
50	1,7940E-02	1,7335E-02	1,3604E-02	1,3445E-02	1,5478E-05	1,7096E-02
60	1,9569E-02	1,8753E-02	1,4303E-02	1,4224E-02	1,6547E-05	1,9154E-02
70	2,0681E-02	2,0537E-02	1,5231E-02	1,5013E-02	1,7377E-05	2,0409E-02
80	2,2307E-02	2,2067E-02	1,5998E-02	1,5920E-02	1,5617E-05	2,2141E-02
90	2,4006E-02	2,3640E-02	1,7009E-02	1,6843E-02	1,5602E-05	2,4490E-02
100	2,5681E-02	2,5585E-02	1,7689E-02	1,7547E-02	1,5826E-05	2,5870E-02

Tabela 4.10 Tempo médio de execução (em segundos) do algoritmo *fn_ qrs_ full_ erf_ spence*

Ordem	**Flag = 1**	**Flag = 2**	**Flag = 3**	**Flag = 4**	**Flag = 5**	**Flag = 6**
10	1,0657E-01	1,0386E-01	1,0587E-01	1,0999E-01	0,0000E+00	1,0583E-01
20	2,2416E-01	2,2060E-01	2,2414E-01	2,3409E-01	0,0000E+00	2,2435E-01
30	3,4172E-01	3,3668E-01	3,4432E-01	3,4758E-01	0,0000E+00	3,4229E-01
40	4,5926E-01	4,5645E-01	4,6290E-01	4,5981E-01	0,0000E+00	4,6190E-01
50	5,7631E-01	5,7132E-01	5,8093E-01	5,7710E-01	0,0000E+00	5,8156E-01
60	6,9341E-01	6,8569E-01	7,0105E-01	6,9531E-01	0,0000E+00	6,9918E-01
70	8,1251E-01	8,0281E-01	8,1973E-01	8,1485E-01	0,0000E+00	8,2154E-01
80	9,3219E-01	9,2095E-01	9,3771E-01	9,3197E-01	0,0000E+00	9,3953E-01
90	1,0491E+00	1,0398E+00	1,0544E+00	1,0499E+00	0,0000E+00	1,0588E+00
100	1,1690E+00	1,1572E+00	1,1784E+00	1,1678E+00	0,0000E+00	1,1764E+00

Na segunda etapa, analisamos o algoritmo proposto por Spence com mais detalhes, e fizemos pequenas modificações, como o uso de vetores ao invés do uso de uma matriz para armazenar variáveis auxiliares, e modificamos a forma como calculamos os pesos de quadratura, de modo que passamos a calculá-los através da equação (3.43). Com essas modificações, implementamos dois algoritmos, um em precisão simples e outro em precisão dupla. Em ambos, o problema de autovalores da matriz tridiagonal simétrica é resolvido através de rotinas da biblioteca LAPACK (*SSTEVD* para precisão simples e *DSTEVD* para precisão dupla), e continuamos calculando as constantes α_k e β_k através da quadratura *função erro*. No decorrer deste trabalho, iremos nos referir as nossas implementações em linguagem Fortran 95 feita na segunda etapa como *fn_qrs_full_erf_slapack* e *fn_qrs_full_erf_dlapack*, para as precisões simples e dupla, respectivamente.

As modificações realizadas reduziram o tempo computacional, mas continuamos não podendo utilizar estes códigos em precisão quádrupla, pois a biblioteca *LAPACK* não está implementada para esta precisão. Na tabela 4.11 apresentamos o tempo médio de 1000 execuções para a obtenção das quadraturas unidimensionais de ordens $N = 10, 20, \ldots, 100$ pelo algoritmo *fn_qrs_full_erf_dlapack*, com os parâmetros $h = 0.004$ e $tol\_erf = 1E-17$ para a quadratura *função erro*.

Tabela 4.11 Tempo médio de execução (em segundos) do algoritmo *fn_ qrs_full_ erf_ dlapack*

Ordem	**Flag = 1**	**Flag = 2**	**Flag = 3**	**Flag = 4**	**Flag = 5**	**Flag = 6**
10	5,0800E-04	4,2600E-04	5,0400E-04	5,0300E-04	0,0000E+00	4,6700E-04
20	6,1200E-04	6,7500E-04	6,1700E-04	5,8600E-04	1,0000E-06	6,2800E-04
30	8,4200E-04	7,9000E-04	7,7300E-04	7,5700E-04	0,0000E+00	7,7200E-04
40	1,0220E-03	1,0100E-03	9,6500E-04	9,6900E-04	0,0000E+00	9,0400E-04
50	1,2010E-03	1,2180E-03	1,1360E-03	1,1710E-03	0,0000E+00	1,0330E-03
60	1,4080E-03	1,4260E-03	1,3720E-03	1,3510E-03	1,0000E-06	1,2800E-03
70	1,5770E-03	1,6220E-03	1,5970E-03	1,5510E-03	0,0000E+00	1,3730E-03
80	1,8000E-03	1,8500E-03	1,6850E-03	1,7650E-03	0,0000E+00	1,5540E-03
90	2,0060E-03	1,9880E-03	1,9430E-03	1,9530E-03	1,0000E-06	1,7380E-03
100	2,2680E-03	2,3260E-03	2,2070E-03	2,2030E-03	0,0000E+00	1,9430E-03

Por fim, na terceira etapa da implementação, modificamos a rotina utilizada para resolver o problema de autovalores (para implementarmos o algoritmo em precisão quádrupla) e buscamos uma quadratura alternativa a quadratura *função erro* que dependa somente de funções intrínsecas da linguagem Fortran 95.

Para a versão em precisão quádrupla, resolvemos o problema de autovalores através da rotina *iter_ qr*, descrita na seção 4.1.3, que permite o cálculo de autovalores para matrizes tridiagonais simétricas em precisão simples, dupla e quádrupla. Esta rotina armazena a matriz J_n através de dois vetores, contendo os valores da diagonal principal e diagonal superior. No decorrer deste trabalho, iremos nos referir a nossa implementação utilizando a rotina *iter_ qr* e a quadratura *função*

erro como *fn_qrs_full_erf*. Tal rotina é capaz de gerar as quadraturas em precisão simples, dupla e quádrupla.

Em termos de tempo computacional, a rotina *fn_qrs_full_erf* (em precisão dupla) apresenta tempo competitivo em comparação com a rotina implementada na segunda etapa. Na tabela 4.12 exibimos o tempo médio de 1000 execuções para a obtenção das quadraturas unidimensionais de ordens $N = 10, 20, \ldots, 100$ do algoritmo *fn_qrs_full_erf* (em precisão dupla), com os parâmetros $h = 0.004$ e $tol\_erf = 1E-17$ para a quadratura *função erro*.

Tabela 4.12 Tempo médio de execução (em segundos) do algoritmo *fn_qrs_full_erf* (em precisão dupla)

Ordem	**Flag = 1**	**Flag = 2**	**Flag = 3**	**Flag = 4**	**Flag = 5**	**Flag = 6**
10	5,2100E-04	4,5100E-04	4,5800E-04	4,1500E-04	0,0000E+00	4,4100E-04
20	6,1400E-04	5,9500E-04	5,9300E-04	6,5800E-04	0,0000E+00	6,1300E-04
30	8,3800E-04	7,8600E-04	7,4900E-04	7,9100E-04	1,0000E-06	7,8500E-04
40	9,8100E-04	9,5400E-04	9,0800E-04	9,8000E-04	0,0000E+00	9,5500E-04
50	1,1910E-03	1,2140E-03	1,0810E-03	1,2200E-03	0,0000E+00	1,1370E-03
60	1,4920E-03	1,3760E-03	1,3350E-03	1,3270E-03	0,0000E+00	1,2870E-03
70	1,6330E-03	1,5360E-03	1,5020E-03	1,5560E-03	1,0000E-06	1,5310E-03
80	1,8240E-03	1,7160E-03	1,7190E-03	1,7750E-03	0,0000E+00	1,7530E-03
90	2,0850E-03	1,9350E-03	1,8880E-03	2,1250E-03	0,0000E+00	1,9790E-03
100	2,2990E-03	2,1880E-03	2,0640E-03	2,2870E-03	1,0000E-06	2,1340E-03

Como alternativa ao esquema de quadratura *função erro*, investigamos o uso do esquema de quadratura *tanh-sinh* [6, 13] na obtenção das constantes α_k e β_k. O princípio utilizado neste esquema é o mesmo da quadratura *função erro*.

Desta forma, na equação (4.76), consideramos

$$g(t) = \tanh\left(\frac{\pi}{2}\sinh(t)\right) \tag{4.77}$$

e

$$g'(t) = \frac{\pi/2\cosh(t)}{\cosh^2\left(\frac{\pi}{2}\sinh(t)\right)}. \tag{4.78}$$

Assim como a quadratura *função erro*, a quadratura *tanh-sinh* aproxima a integral através de uma soma infinita, de modo que truncamos esta soma infinita dada pela equação (4.76) quando $w_j < tol\_tanh - sinh$, em que $tol\_tanh - sinh$ é uma tolerância que nós determinamos. Com isso, para a implementação desta quadratura, também são necessários a determinação de dois parâmetros: h e $tol\_tanh - sinh$, responsáveis pelo passo de quadratura e critério de parada, respectivamente. Novamente, caso o intervalo de integração (a, b) seja outro que não $(-1, 1)$, basta fazermos um mapeamento do intervalo (a, b) para o intervalo $(-1, 1)$ [13]. A vantagem do uso da quadratura *tanh-sinh* para a linguagem Fortran 95 está no fato de que as funções tanh, sinh e cosh são todos intrínsecos da linguagem, e assim, podemos utilizar este esquema de quadratura independente do compilador utilizado.

Utilizando a quadratura *tanh-sinh*, implementamos em linguagem Fortran 95 um algoritmo que também é capaz de gerar as quadraturas em precisão simples, dupla e quádrupla com um baixo tempo computacional. No decorrer deste trabalho, iremos nos referir a essa implementação em linguagem Fortran 95 como *fn_qrs_full_tanh_sinh*. A tabela 4.13 apresenta o tempo médio de 1000 execuções para a obtenção das quadraturas unidimensionais de ordens $N = 10, 20, \ldots, 100$ do

algoritmo *fn_ qrs_full_ tanh_ sinh* (em precisão dupla), com parâmetros $h = 0.004$ e $tol\_tanh - sinh = 1E - 17$.

Tabela 4.13 Tempo médio de execução (em segundos) algoritmo *fn_ qrs_full_ tanh_ sinh* (em precisão dupla)

Ordem	**Flag = 1**	**Flag = 2**	**Flag = 3**	**Flag = 4**	**Flag = 5**	**Flag = 6**
10	3,6200E-04	2,9300E-04	3,1300E-04	2,9400E-04	0,0000E+00	2,9500E-04
20	3,8500E-04	3,7900E-04	3,9600E-04	3,9100E-04	0,0000E+00	4,5100E-04
30	4,6500E-04	4,6200E-04	4,8500E-04	5,1900E-04	0,0000E+00	5,5800E-04
40	5,7200E-04	5,6700E-04	5,8700E-04	6,5400E-04	1,0000E-06	6,4400E-04
50	6,7600E-04	6,8600E-04	6,9700E-04	7,2900E-04	0,0000E+00	7,4900E-04
60	8,0600E-04	7,8900E-04	8,1200E-04	8,3200E-04	0,0000E+00	9,1100E-04
70	9,2100E-04	9,1000E-04	9,4000E-04	9,7600E-04	0,0000E+00	1,0320E-03
80	1,0570E-03	1,0500E-03	1,0650E-03	1,1580E-03	1,0000E-06	1,1380E-03
90	1,1940E-03	1,1880E-03	1,2140E-03	1,3040E-03	0,0000E+00	1,3330E-03
100	1,3410E-03	1,3370E-03	1,3580E-03	1,5700E-03	0,0000E+00	1,4310E-03

Em ambos esquemas de quadraturas, *função erro* e *tanh-sinh*, utilizamos os mesmos parâmetros, e testamos outros valores para h, tais como $h = 0.032$, $h = 0.016$, $h = 0.002$, $h = 0.001$, $h = 2^{-5}$, $h = 2^{-6}$ e $h = 2^{-7}$, sendo que obtivemos o melhor desempenho com o mesmo parâmetro proposto por Spence, $h = 0.004$. Quanto ao valor para a tolerância, tol_erf e $tol\_tanh - sinh$, testamos outros valores: $tol = 1E - 15$, $tol = 1E - 16$, $tol = 1E - 18$ e $tol = \varepsilon_M$, e obtivemos desempenho semelhante em todos os casos. Com isso, estabelecemos o valor do passo como $h = 0.004$ e tolerância $tol = \varepsilon_M$, pois obteve desempenho satisfatório em precisão dupla, além de ser um valor adequado para precisão quádrupla.

Apesar de, do ponto de vista teórico, a abordagem via polinômios ortogonais ser capaz de gerar nós e pesos de quadratura para ordens arbitrárias, notamos que na prática, para cada *flag* existe uma ordem máxima na qual é possível gerar os nós e pesos da respectiva quadratura. Notamos também que estas ordens máximas estão associadas, principalmente, ao valor de h utilizado tanto na quadratura *função erro* quanto na quadratura *tanh-sinh*. Na tabela 4.14, apresentamos as ordens máximas em precisão dupla, com parâmetros $h = 0.004$ e $tol = \varepsilon_M$, para as rotinas descritas anteriormente, juntamente com o algoritmo apresentado por Spence implementado no Matlab R2018b. Quanto as ordens máximas para as demais precisões, em precisão simples somente a $flag = 5$ é capaz de gerar a quadratura com ordem $N \geq 50$, e em precisão quádrupla, todas as *flags* são capazes de gerar quadraturas com ordem $N \geq 2200$. Em particular, a $flag = 5$ possui limitantes mais altos que as demais, pois sua geração não depende das quadraturas *função erro* ou *tanh-sinh*, uma vez que seus pesos são todos iguais e seus nós são obtidos através da função *cosseno*. Caso seja utilizada uma ordem de quadratura superior as ordens máximas, o programa retorna para os nós e pesos o valor NaN (*not a number*).

Tabela 4.14 Ordem máxima na qual é possível gerar a quadratura em precisão dupla com parâmetros $h = 0.004$ e $tol = \varepsilon_M$

Flag	**Matlab R2018b**	**fn_qrs_full_erf_spence**	**fn_qrs_full_erf_dlapack**	**fn_qrs_full_tanh_sinh**
1	359	341	341	341
2	226	215	215	215
3	180	171	171	171
4	269	256	256	256
5	25000	25000	25000	25000
6	215	205	205	205

Na proxima seção, discutimos duas formas de realizar o acoplamento das quadraturas unidimensionais para as variáveis polar e azimutal.

4.2.5 Acoplamento das quadraturas polar e azimutal

Nesta seção, apresentamos dois tipos de acoplamentos para os esquemas de quadratura QR, que são o acoplamento quadrangular [70] e o acoplamento triangular [43]. Ambos acoplamentos são gerados através do produto de um conjunto de quadratura para a variável polar por um conjunto de quadratura para a variável azimutal, discutidos nas seções 4.2.3.1 e 4.2.3.2, respectivamente, e se diferenciam pela forma como são combinados os graus das quadraturas unidimensionais utilizadas. Dessa forma, os diferentes esquemas resultam, por exemplo, em diferentes números de direções discretas por octante na esfera unitária. Além destes acoplamentos, outras formas podem ser utilizadas, como as apresentadas por Abu-Shumays [2].

4.2.5.1 Quadratura QR Quadrangular

Para obtermos este acoplamento, as ordens das quadraturas unidimensionais para as variáveis polar e azimutal utilizadas necessitam ser iguais. Dessa forma, definida a ordem N_θ da quadratura polar desejada, definimos a ordem da quadratura azimutal, N_ϕ, de modo que $N_\phi = N_\theta$. Obtida a quadratura unidimensional para a variável polar, θ, e a quadratura para a variável azimutal, ϕ, definimos o conjunto de pontos da quadratura QR quadrangular a partir das equações (2.3) e (2.4):

$$\mu_{i,j} = \sin(\theta_i)\cos(\phi_j), \tag{4.79}$$

$$\eta_{i,j} = \sin(\theta_i)\sin(\phi_j) \tag{4.80}$$

e

$$W_{i,j} = w_i^p w_j^a, \tag{4.81}$$

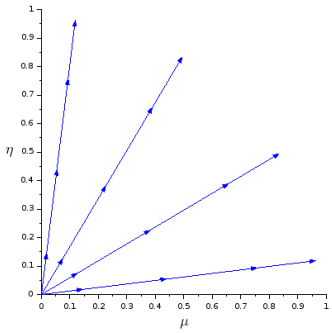
com $i, j = 1, \ldots, N_\theta$, $W_{i,j}$ o peso da quadratura produto, w_i^p e w_j^a os pesos das quadraturas unidimensionais para as variáveis polar e azimutal, respectivamente.

Uma vez que na seção 4.2.3.2 apresentamos o total de cinco conjuntos de quadratura azimutal, dados pelas *flags* 1, 2, 3, 5 e 6, temos o total de cinco esquemas de quadratura QR quadrangular, diferenciando-se uns dos outros pela quadratura azimutal utilizada. Na tabela 4.15, apresentamos a notação utilizada para cada esquema, em que N_θ representa a ordem da quadratura polar utilizada e m indica que tais esquemas são gerados numericamente através da rotina *fn_ qrs_full_ tanh_ sinh*.

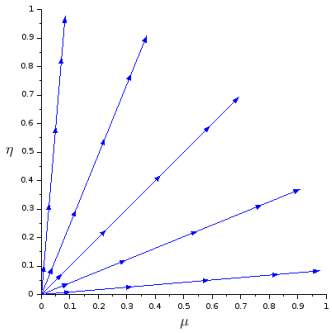
Tabela 4.15 Notação quadratura QR quadrangular

Notação	**Flag Azimutal**
$QRS45m\_Q_{N_\theta}$	1
$QRA45m\_Q_{N_\theta}$	2
$QRS90m\_Q_{N_\theta}$	3
$QRJ45m\_Q_{N_\theta}$	5
$QRJ90m\_Q_{N_\theta}$	6

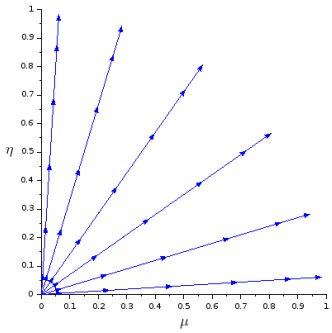
Devido a forma como este acoplamento é realizado, quando as direções são projetadas no plano $\mu\eta$ observamos que há direções que ficam sobrepostas, como pode ser observado nas figuras 4.6, 4.7, 4.9, 4.8 e 4.10. Este comportamento ocorre pelo mesmo motivo da quadratura $P_N T_N$, isto é, se deve ao fato de todos os níveis polares utilizarem a quadratura azimutal de mesma ordem. Por fim, o número de direções discretas por octante da esfera unitária é dado por $M = N_\theta^2$.



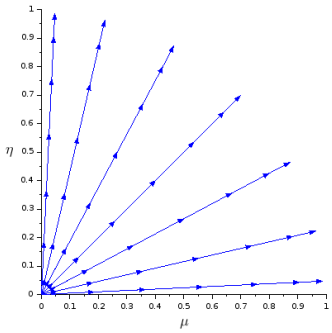
(a) $QRS45m\_Q_4$



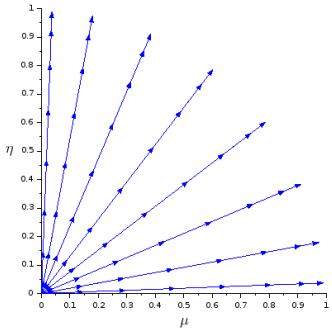
(b) $QRS45m\_Q_5$



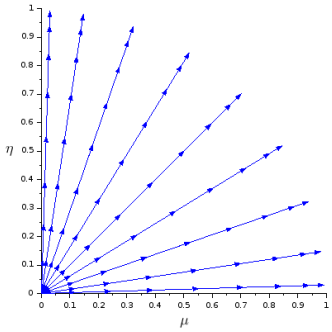
(c) $QRS45m\_Q_6$



(d) $QRS45m\_Q_7$

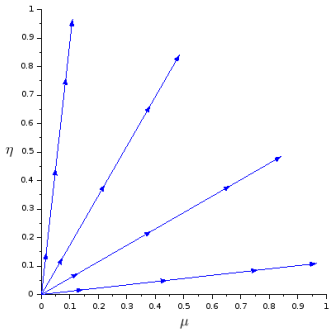


(e) $QRS45m\_Q_8$

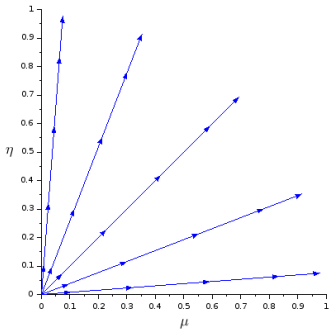


(f) $QRS45m\_Q_9$

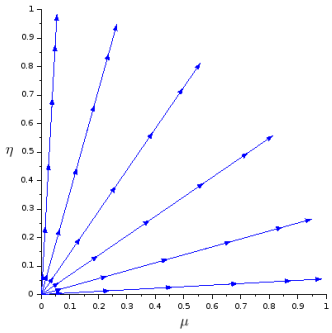
Figura 4.6 Distribuição das direções discretas quadratura $QRS45m\_Q_{N_\theta}$ no plano $\mu\eta$



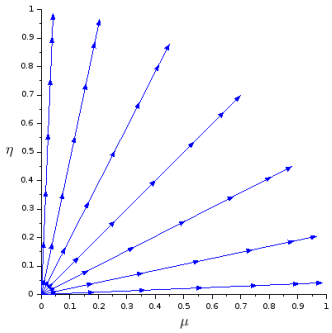
(a) $QRA45m\_Q_4$



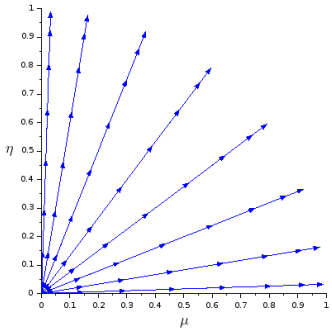
(b) $QRA45m\_Q_5$



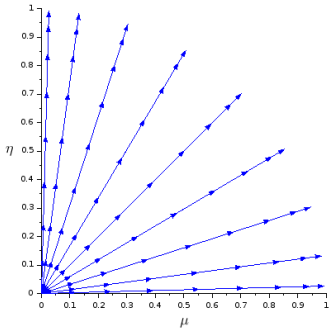
(c) $QRA45m\_Q_6$



(d) $QRA45m\_Q_7$

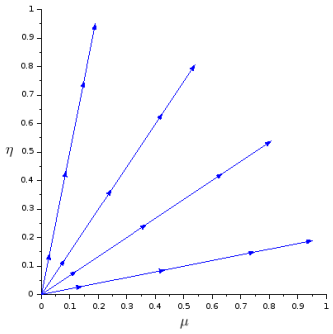


(e) $QRA45m\_Q_8$

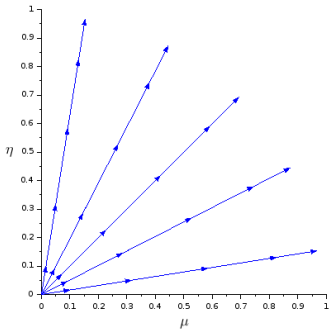


(f) $QRA45m\_Q_9$

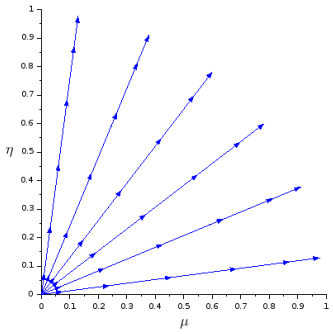
Figura 4.7 Distribuição das direções discretas quadratura $QRA45m\_Q_{N_\theta}$ no plano $\mu\eta$



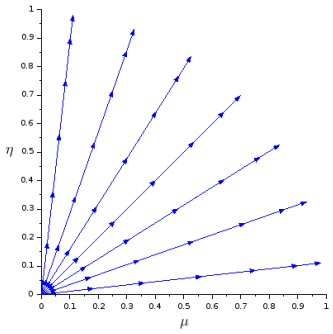
(a) $QRJ45m\_Q_4$



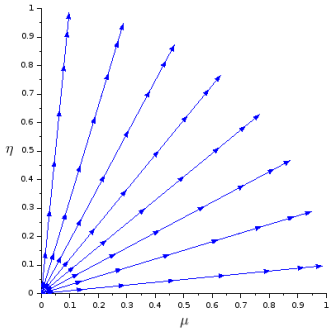
(b) $QRJ45m\_Q_5$



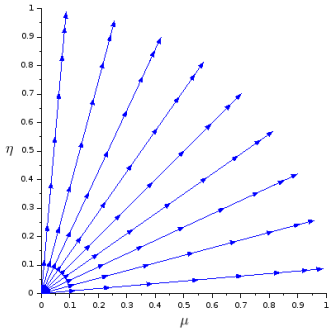
(c) $QRJ45m\_Q_6$



(d) $QRJ45m\_Q_7$

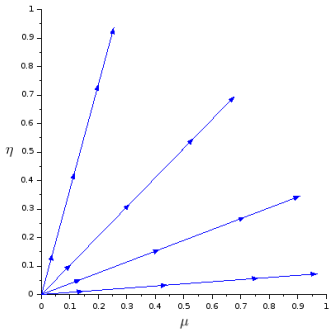


(e) $QRJ45m\_Q_8$

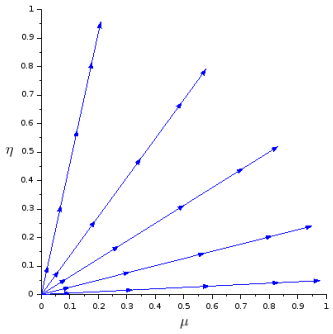


(f) $QRJ45m\_Q_9$

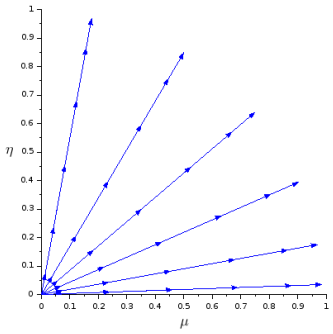
Figura 4.8 Distribuição das direções discretas quadratura $QRJ45m\_Q_{N_\theta}$ no plano $\mu\eta$



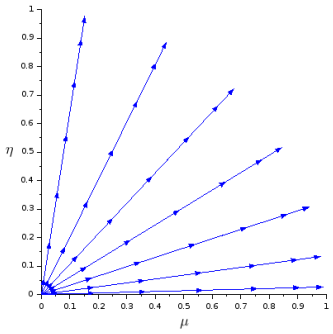
(a) $QRS90m\_Q_4$



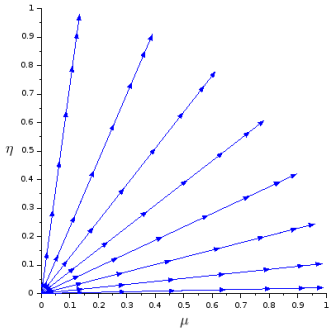
(b) $QRS90m\_Q_5$



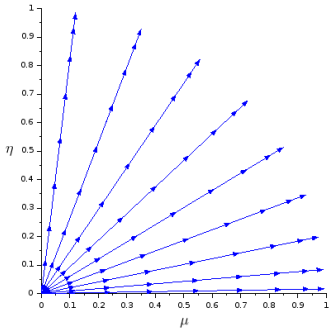
(c) $QRS90m\_Q_6$



(d) $QRS90m\_Q_7$

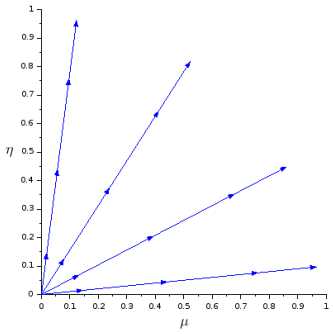


(e) $QRS90m\_Q_8$

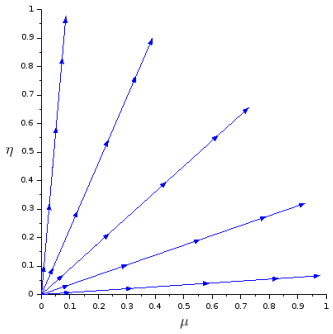


(f) $QRS90m\_Q_9$

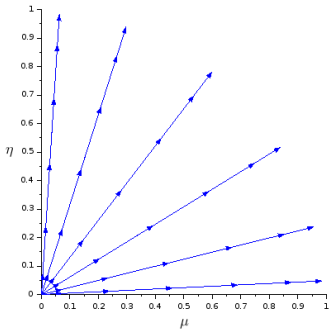
Figura 4.9 Distribuição das direções discretas quadratura $QRS90m\_Q_{N_\theta}$ no plano $\mu\eta$



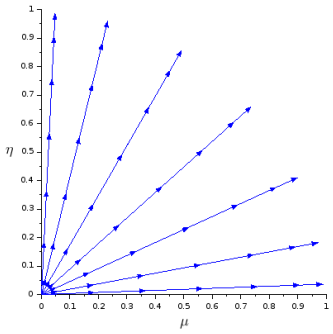
(a) $QRJ90m\_Q_4$



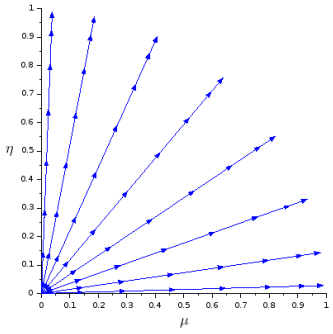
(b) $QRJ90m\_Q_5$



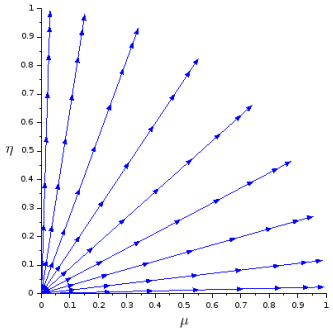
(c) $QRJ90m\_Q_6$



(d) $QRJ90m\_Q_7$



(e) $QRJ90m\_Q_8$



(f) $QRJ90m\_Q_9$

Figura 4.10 Distribuição das direções discretas quadratura $QRJ90m\_Q_{N_\theta}$ no plano $\mu\eta$

4.2.5.2 Quadratura QR Triangular

Para obtermos este acoplamento, relacionamos as ordens das quadraturas unidimensionais de forma semelhante à quadratura $P_N T_N S_N$, isto é, definidos os ξ_i níveis polares, $\xi_i = \cos(\theta_i)$, para uma certa ordem N_θ, no primeiro nível polar, ξ_1, utilizamos a quadratura azimutal de ordem N_θ; no segundo nível polar, ξ_2, utilizamos a quadratura azimutal de ordem $N_\theta - 1$; no i-ésimo nível polar, ξ_i, utilizamos a quadratura azimutal de ordem ordem $N_\theta + 1 - i$. Assim como na seção 4.1.2, consideramos o primeiro nível polar, ξ_1, aquele mais próximo ao centro da esfera unitária, enquanto o último nível polar, ξ_{N_θ}, é o mais próximo da borda da esfera unitária.

Obtida a quadratura unidimensional para a variável polar, θ, e a quadratura para a variável azimutal, ϕ, definimos o conjunto de pontos da quadratura QR triangular a partir das equações (2.3) e (2.4):

$$\mu_{i,j} = \sin(\theta_i)\cos(\phi_j), \tag{4.82}$$

$$\eta_{i,j} = \sin(\theta_i)\sin(\phi_j) \tag{4.83}$$

e

$$W_{i,j} = w_i^p w_j^a, \tag{4.84}$$

com $i = 1, \ldots, N_\theta$, $j = 1, \ldots, N_\theta + 1 - i$, $W_{i,j}$ o peso da quadratura produto, w_i^p e w_j^a os pesos das quadraturas unidimensionais para as variáveis polar e azimutal, respectivamente.

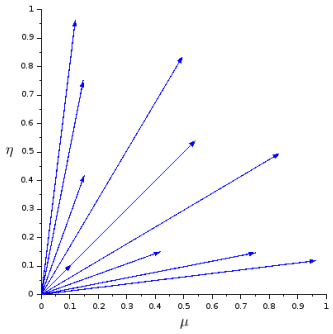
Assim como na quadratura QR quadrangular, temos o total de cinco esquemas de quadratura QR triangular, diferenciando-se uns dos outros pela quadratura azimutal utilizada. Na tabela 4.16, apresentamos a notação utilizada para

cada esquema, em que N_θ representa a ordem da quadratura polar, sendo que tais esquemas são gerados numericamente através da rotina *fn_qrs_full_tanh_sinh*.

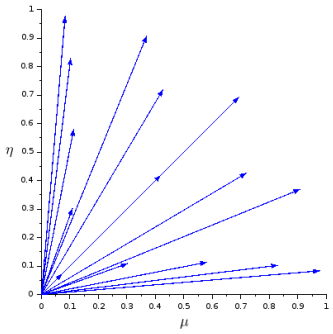
Tabela 4.16 Notação quadratura QR triangular

Notação	**Flag Azimutal**
$QRS45m\_T_{N_\theta}$	1
$QRA45m\_T_{N_\theta}$	2
$QRS90m\_T_{N_\theta}$	3
$QRJ45m\_T_{N_\theta}$	5
$QRJ90m\_T_{N_\theta}$	6

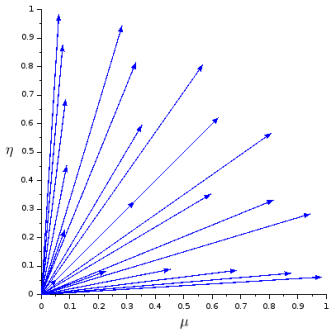
Devido a forma como o acoplamento triangular é realizado, quando as direções são projetadas no plano $\mu\eta$, observamos que as quadraturas $QRS45m\_T_{N_\theta}$, $QRA45m\_T_{N_\theta}$ e $QRJ45m\_T_{N_\theta}$ (que possuem simetria em relação a $\phi = \pi/4$) possuem uma direção que fica sobreposta (exatamente em $\phi = \pi/4$), como pode ser visto nas figuras 4.11, 4.12 e 4.13. Isto ocorre pois $\phi = \pi/4$ sempre é um nó de quadratura para N_θ ímpar. Já para as quadraturas $QRS90m\_T_{N_\theta}$ e $QRJ90m\_T_{N_\theta}$ (que não possuem simetria azimutal), não há direções sobrepostas, como pode ser visto nas figuras 4.14 e 4.15. Este comportamento decorre de em cada nível polar utilizar uma ordem de quadratura azimutal diferente. Por fim, o número de direções discretas por octante da esfera unitária é dado por $M = N_\theta(N_\theta + 1)/2$.



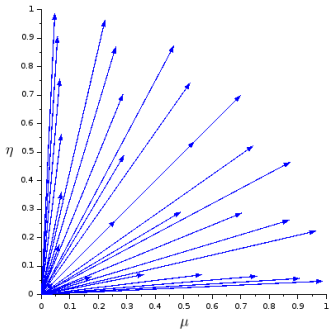
(a) $QRS45m\_T_4$



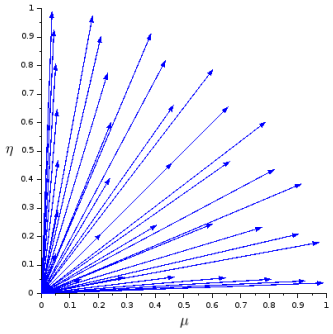
(b) $QRS45m\_T_5$



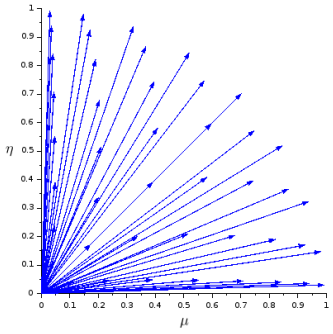
(c) $QRS45m\_T_6$



(d) $QRS45m\_T_7$

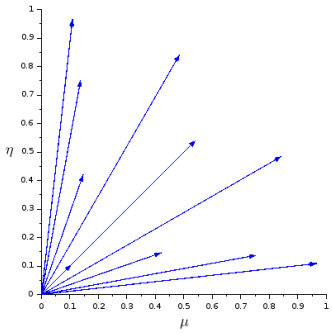


(e) $QRS45m\_T_8$

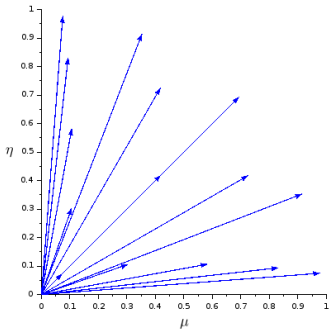


(f) $QRS45m\_T_9$

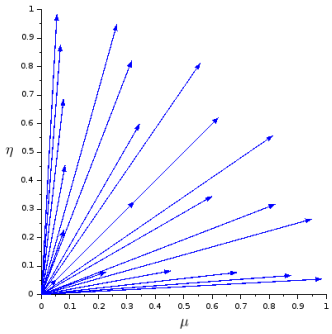
Figura 4.11 Distribuição das direções discretas quadratura $QRS45m\_T_{N_\theta}$ no plano $\mu\eta$



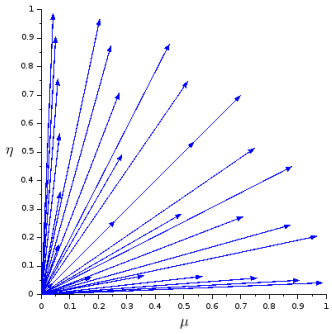
(a) $QRA45m\_T_4$



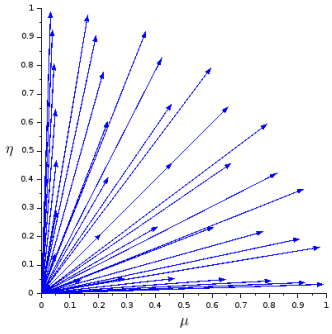
(b) $QRA45m\_T_5$



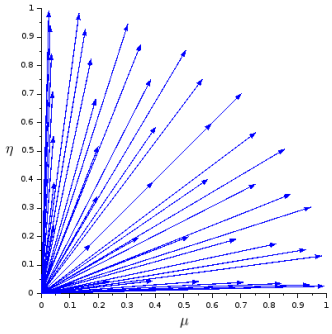
(c) $QRA45m\_T_6$



(d) $QRA45m\_T_7$

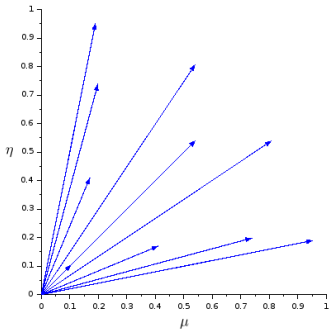


(e) $QRA45m\_T_8$

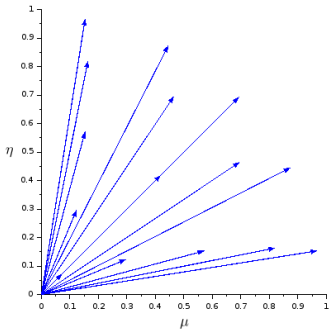


(f) $QRA45m\_T_9$

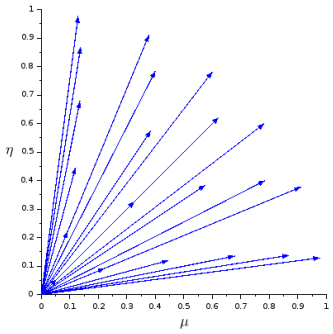
Figura 4.12 Distribuição das direções discretas quadratura $QRA45m\_T_{N\theta}$ no plano $\mu\eta$



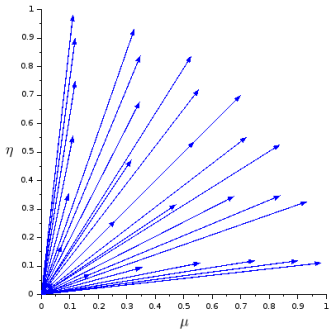
(a) $QRJ45m\_T_4$



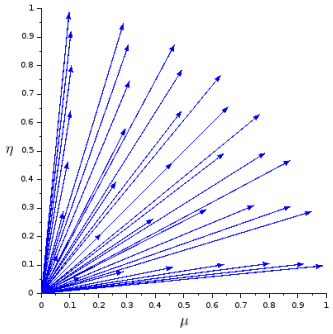
(b) $QRJ45m\_T_5$



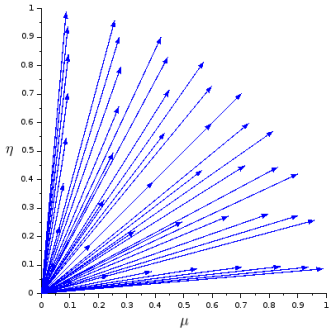
(c) $QRJ45m\_T_6$



(d) $QRJ45m\_T_7$

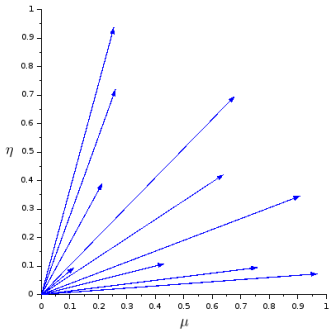


(e) $QRJ45m\_T_8$

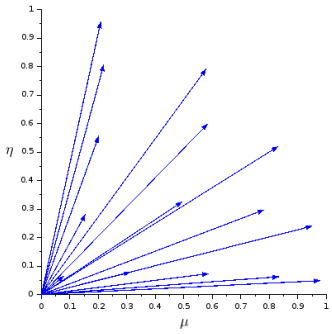


(f) $QRJ45m\_T_9$

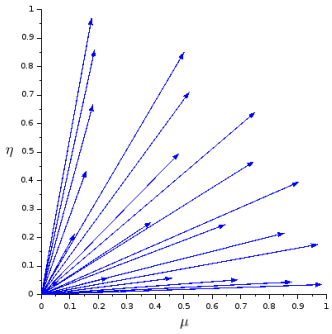
Figura 4.13 Distribuição das direções discretas quadratura $QRJ45m\_T_{N_\theta}$ no plano $\mu\eta$



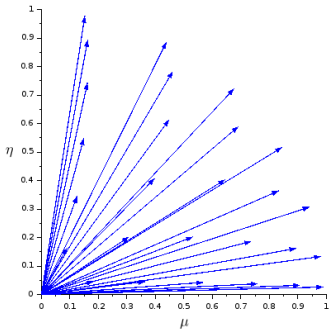
(a) $QRS90m\_T_4$



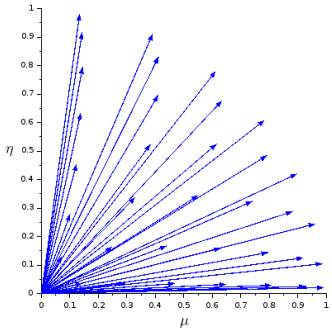
(b) $QRS90m\_T_5$



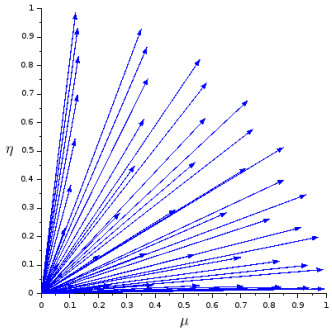
(c) $QRS90m\_T_6$



(d) $QRS90m\_T_7$

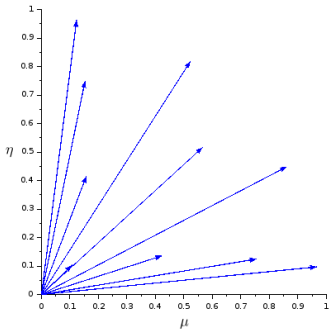


(e) $QRS90m\_T_8$

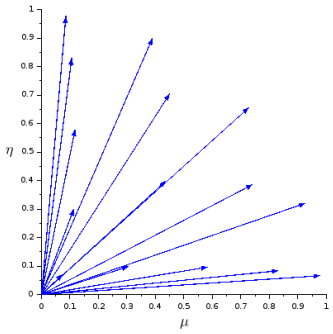


(f) $QRS90m\_T_9$

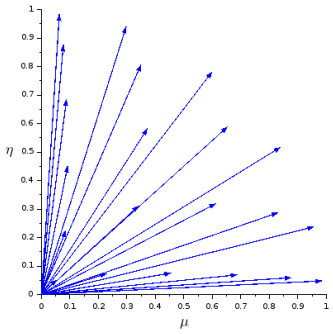
Figura 4.14 Distribuição das direções discretas quadratura $QRS90m\_T_{N_\theta}$ no plano $\mu\eta$



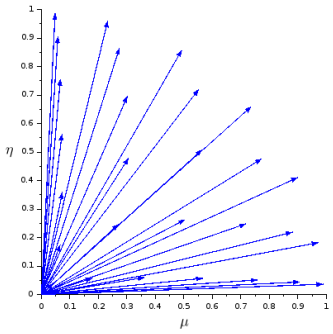
(a) $QRJ90m\_T_4$



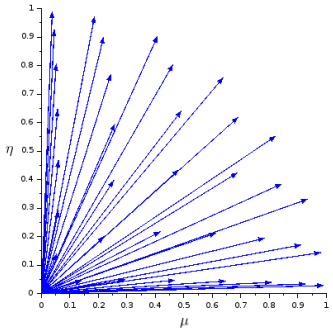
(b) $QRJ90m\_T_5$



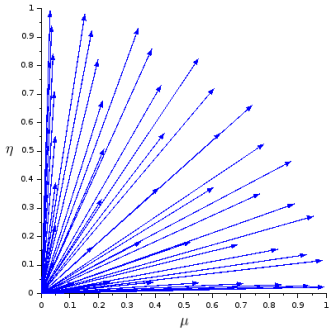
(c) $QRJ90m\_T_6$



(d) $QRJ90m\_T_7$



(e) $QRJ90m\_T_8$



(f) $QRJ90m\_T_9$

Figura 4.15 Distribuição das direções discretas quadratura $QRJ90m\_T_{N_\theta}$ no plano $\mu\eta$

Na próxima seção deste capítulo, apresentamos alguns resultados importantes no que diz respeito à estimativas e comportamento do erro nas aproximações de integrais multidimensionais por quadraturas.

4.3 Estimativa do Erro de Truncamento de Quadraturas Produto

Iniciamos este estudo apresentando uma estimativa do erro de truncamento de quadraturas produto em um retângulo, de acordo com Stroud e Secrest [71]. Considere

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x,y)dxdy \approx \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} A_i B_j f(x_i,y_j), \tag{4.85}$$

em que A_i e x_i representam, respectivamente, os pesos e nós de uma regra de quadratura para o intervalo $[a_1,b_1]$, e B_j e y_j representam, respectivamente, os pesos e nós de uma regra de quadratura para o intervalo $[a_2,b_2]$. Assuma que, para todas as funções $g_1(x)$ em uma certa classe W_x, temos

$$\left|\int_{a_1}^{b_1} g_1(x)dx - \sum_{i=1}^{m_1} A_i g_1(x_i)\right| \leq c_x, \tag{4.86}$$

e para todas as funções $g_2(y)$ de outra classe W_y, temos

$$\left|\int_{a_2}^{b_2} g_2(y)dx - \sum_{j=1}^{m_2} B_j g_2(y_j)\right| \leq c_y. \tag{4.87}$$

Também assuma que para todo o x,y no retângulo $[a_1,b_1]\times[a_2,b_2]$, a função $f(x,y)$ pertence a ambas classes, W_x e W_y.

Então,

$$\left|\int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x,y)dxdy - \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} A_i B_j f(x_i,y_j)\right| = \left|\int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x,y)dxdy - \int_{a_2}^{b_2}\sum_{i=1}^{m_1} A_i f(x_i,y)dy\right.$$

$$
\begin{aligned}
&+\sum_{i=1}^{m_1} A_i \int_{a_2}^{b_2} f(x_i, y)dy - \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} A_i B_j f(x_i, y_j)\Bigg| \\
&\leq \int_{a_2}^{b_2} \left| \int_{a_1}^{b_1} f(x, y)dx - \sum_{i=1}^{m_1} A_i f(x_i, y)\right| dy \\
&+\sum_{i=1}^{m_1} |A_i| \left| \int_{a_2}^{b_2} f(x_i, y)dy - \sum_{j=1}^{m_2} B_j f(x_i, y_j)\right| \\
&\leq (b_2 - a_2)c_x + c_y \sum_{i=1}^{m_1} |A_i|. \qquad (4.88)
\end{aligned}
$$

De modo análogo, podemos estimar

$$
\left| \int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x, y)dxdy - \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} A_i B_j f(x_i, y_j)\right| \leq (b_1 - a_1)c_y + c_x \sum_{j=1}^{m_2} |B_j|. \qquad (4.89)
$$

Já o resultado a seguir nos diz quando uma quadratura produto integra exatamente uma função $h(x, y)$ [23, 43]: sejam Q_n uma quadratura unidimensional com n pontos capaz de integrar exatamente a função $f(x)$ sobre o intervalo $\mathbb{A}$, e Q_m uma quadratura unidimensional com m pontos capaz de integrar exatamente a função $g(y)$ sobre o intervalo $\mathbb{B}$. Então, a quadratura produto $Q_n \times Q_m$ integra exatamente a função $h(x, y) = f(x)g(y)$ sobre a região $\mathbb{A} \times \mathbb{B}$.

Para o caso d-dimensional, escrevemos a quadratura produto como [43]

$$
Q^d[f] = \sum_{i_1=1}^{n_1} w_{i_1}^1 \sum_{i_2=1}^{n_2} w_{i_2}^2 \cdots \sum_{i_d=1}^{n_d} w_{i_d}^d f(x_{i_1}^1, x_{i_2}^2, \cdots, x_{i_d}^d), \qquad (4.90)
$$

correspondendo a seguinte integral:

$$
I[f] = \int_{a_1}^{b_1}\int_{a_2}^{b_2} \cdots \int_{a_d}^{b_d} f(x_1, x_2, \cdots, x_d)dx_1dx_2\cdots dx_d, \qquad (4.91)
$$

em que

- d é a dimensão;

- n_j é o número de pontos da j-ésima quadratura unidimensional, $j = 1, 2, \ldots, d$;

- w_i^j e x_i^j são os i-ésimos pesos e nós da j-ésima quadratura unidimenional, $i = 1, 2, \ldots, n_j$, respectivamente.

Haber [40] mostra uma estimativa para o erro de truncamento da quadratura produto sobre $[0, 1]^d$, mas que pode ser aplicada para um intervalo de integração geral. Assumindo que $n_j = \mathcal{O}(n)$ para todo $j = 1, 2, \ldots, n_j$, temos que o número total de pontos na quadratura produto é $M_t = \mathcal{O}(n^d)$, e assumindo que o erro de truncamento da j-ésima quadratura unidimensinal é

$$E_{n_j}^j[f] = \int_{a_j}^{b_j} f(c_1, c_2, \ldots, x_j, \ldots, c_d)dx_j - \sum_{i_j=1}^{n_j} w_{i_j}^j f(c_1, c_2, \ldots, x_{i_j}^j, \ldots, c_d) = \mathcal{O}(n_j^{p_j}), \tag{4.92}$$

para todo $j = 1, 2, \ldots, d$, em que

- $c_k \in (a_k, b_k)$, $k = 1, 2, \ldots, j-1, j+1, \ldots, d$, é uma constante arbitrária;

- p_j é a ordem de convergência do erro de truncamento da j-ésima quadratura unidimensional;

temos que o erro de truncamento da quadratura produto converge como

$$E_{M_t}^Q[f] = I[f] - Q^d[f] = \mathcal{O}(M_t^{P/d}), \tag{4.93}$$

com $P = max_j p_j$. Desta forma, a teoria de Haber indica que a ordem de convergência do erro de truncamento de quadraturas produto é determinado por dois fatores: a ordem de convergência de cada quadratura unidimensional e a dimensão da integral.

Na próxima seção deste capítulo propomos um conjunto de quadratura produto para o caso em que temos uma singularidade na direção azimutal em $\phi = \phi_0$.

4.4 Quadratura com Singularidade Azimutal em $\phi = \phi_0$

Conforme discutido nas seções 3.3 e 4.3, a existência de singularidades tem extrema relevância no comportamento assintótico do erro na aproximação das integrais na esfera unitária por quadraturas multidimensionais. Com isto em mente, nesta seção propomos um esquema de quadratura produto para o caso em que temos uma singularidade na direção azimutal em $\phi = \phi_0$. Nossa proposta é baseada na quadratura QR discutida na seção anterior. Assim como proposto por Abu-Shumays, construímos esta quadratura para o octante principal da esfera unitária, de modo que consideramos o intervalo de definição para o ângulo polar $\theta \in [0, \pi/2]$, e para o ângulo azimutal o intervalo $\phi \in [0, \pi/2]$.

No que diz respeito a ϕ_0, tomamos o intervalo de definição $\phi_0 \in (0, \pi/2) \setminus \{\pi/4\}$, pois se a singularidade for $\phi_0 = 0$ ou $\phi_0 = \pi/2$, todas as quadraturas QR discutidas anteriormente tratam do problema, visto que estes valores não são nós da quadratura associada a variável azimutal para nenhuma $flag$. Também desconsideramos o caso $\phi_0 = \pi/4$, pois basta considerar uma quadratura para a variável azimutal com simetria em relação a $\phi = \pi/4$ ($flags$ 1, 2 e 5) de ordem N_ϕ par.

Dessa forma, buscamos uma aproximação por quadratura da integral

$$\int_0^{\pi/2} \int_0^{\pi/2} f(\theta, \phi) sin(\theta) d\phi d\theta \approx \sum_{i=1}^{N_\theta} \sum_{j=1}^{N_\phi} w_i^p w_j^a f(\theta_i, \phi_j), \tag{4.94}$$

em que a função $f(\theta, \phi)$ possui um ponto de singularidade azimutal em $\phi = \phi_0$.

Iniciamos este estudo pelo tratamento da variável azimutal. Buscamos então, inicialmente, uma aproximação por quadratura da integral

$$\int_0^{\pi/2} \cos^l(\phi) \sin^m(\phi) d\phi \approx \sum_{j=1}^{N_\phi} w_j^a \cos^l(\phi_j) \sin^m(\phi_j), \tag{4.95}$$

de modo que, $\phi_j \neq \phi_0$, $j = 1, \ldots, N_\phi$, e $\phi_0 \in (0, \pi/2) \setminus \{\pi/4\}$. Reescrevemos a equação (4.95) na forma

$$\int_0^{\pi/2} \cos^l(\phi) \sin^m(\phi) d\phi = \int_0^{\phi_0} \cos^l(\phi) \sin^m(\phi) d\phi + \int_{\phi_0}^{\pi/2} \cos^l(\phi) \sin^m(\phi) d\phi \tag{4.96}$$

$$\approx \sum_{j=1}^{N_{\phi_1}} w_j^a \cos^l(\phi_j) \sin^m(\phi_j) + \sum_{k=1}^{N_{\phi_2}} w_k^a \cos^l(\phi_k) \sin^m(\phi_k), \tag{4.97}$$

em que

$$\int_0^{\phi_0} \cos^l(\phi) \sin^m(\phi) d\phi \approx \sum_{j=1}^{N_{\phi_1}} w_j^a \cos^l(\phi_j) \sin^m(\phi_j) \tag{4.98}$$

e

$$\int_{\phi_0}^{\pi/2} \cos^l(\phi) \sin^m(\phi) d\phi \approx \sum_{k=1}^{N_{\phi_2}} w_k^a \cos^l(\phi_k) \sin^m(\phi_k). \tag{4.99}$$

Para o tratamento das integrais nas equações (4.98) e (4.99) fazemos a restrição de que $l = m$. Dessa forma, para a equação (4.98), aplicamos a regra

$$\int_0^{\phi_0} \cos^m(\phi) \sin^m(\phi) d\phi = \int_0^{\phi_0} \left(\frac{1}{2} \sin(2\phi) \right)^m d\phi \tag{4.100}$$

$$= \int_0^c \frac{t^m dt}{\sqrt{1 - 4t^2}} \tag{4.101}$$

$$\approx \sum_{j=1}^{N_{\phi_1}} w_j^a \cos^l(\phi_j) \sin^m(\phi_j), \tag{4.102}$$

em que fizemos a substituição $t = \frac{1}{2}\sin(2\phi)$, e $c = \frac{1}{2}\sin(2\phi_0)$. A função peso é dada por

$$\omega_5(t) = \frac{1}{\sqrt{1-4t^2}}, \tag{4.103}$$

que é positiva sobre o domínio de integração. Como a função peso não possui singularidade no intervalo de integração, os produtos internos

$$\langle t\pi_j(t), \pi_j(t)\rangle = \int_0^c t\pi_j^2(t)\omega_5 dt \tag{4.104}$$

e

$$\langle \pi_j(t), \pi_j(t)\rangle = \int_0^c \pi_j^2(t)\omega_5 dt \tag{4.105}$$

são calculados através da quadratura *tanh-sinh* ou da quadratura *função erro*. Uma vez gerado o conjunto de abscissas, $\{t_j\}$, podemos reescrevê-lo em termos do ângulo azimutal, ϕ, notando que

$$\phi_j = \frac{1}{2}\arcsin(2t_j). \tag{4.106}$$

De maneira análoga, para a equação (4.99), aplicamos a regra

$$\int_{\phi_0}^{\pi/2} \cos^m(\phi)\sin^m(\phi)d\phi = \int_{\phi_0}^{\pi/2}\left(\frac{1}{2}\sin(2\phi)\right)^m d\phi \tag{4.107}$$

$$= \int_0^{\pi/2-\phi_0}\left(\frac{1}{2}\sin(2\gamma)\right)^m d\phi \tag{4.108}$$

$$= \int_0^c \frac{t^m dt}{\sqrt{1-4t^2}} \tag{4.109}$$

$$\approx \sum_{k=1}^{N_{\phi_2}} w_j^a \cos^l(\phi_k)\sin^m(\phi_k), \tag{4.110}$$

em que fizemos as substituições $\phi = \gamma + \phi_0$ e $t = \frac{1}{2}\sin(2\gamma)$, e $c = \frac{1}{2}\sin(2\phi_0)$. Note que, os produtos internos que necessitam ser calculados são iguais ao do caso anterior.

Uma vez gerado o conjunto de abscissas, $\{t_k\}$, podemos reescrevê-lo em termos do ângulo azimutal, ϕ, notando que

$$\phi_k = \frac{\pi}{2} - \frac{1}{2}\arcsin(2t_k). \tag{4.111}$$

Dessa forma, os nós e pesos da quadratura unidimensional na variável azimutal para o caso de uma singularidade em $\phi = \phi_0$, são compostos de duas partes: uma parte associada ao intervalo $[0, \phi_0)$ e outra parte associada ao intervalo $(\phi_0, \pi/2]$. Os nós e pesos para cada intervalo são obtidos conforme discutido acima.

Quanto a variável polar, o tratamento é o mesmo que para a quadratura QR, tratada nas seções 4.2.1 e 4.2.3.1, uma vez que não estamos considerando singularidade na variável polar. Acoplamos a quadratura associada a variável polar e a variável azimutal da seguinte forma: definidos os N_θ níveis polares, em cada nível polar, $\xi_i = cos(\theta_i)$, utilizamos a mesma ordem de quadratura azimutal $N_\phi = N_{\phi_1} + N_{\phi_2}$, ou seja,

$$\left(\int_0^{\pi/2} \sin^{l+m}(\theta)\sin(\theta)d\theta\right)\left(\int_0^{\pi/2} \cos^l(\phi)\sin^m(\phi)d\phi\right) \approx$$
$$\left(\sum_{i=1}^{N_\theta} w_i^p \sin^{l+m}(\theta_i)\right)\left(\sum_{j=1}^{N_{\phi_1}} w_j^a \cos^l(\phi_j)\sin^m(\phi_j) + \sum_{k=1}^{N_{\phi_2}} w_k^a \cos^l(\phi_k)\sin^m(\phi_k)\right). \tag{4.112}$$

Obtida a quadratura associada a variável polar e a quadratura associada a variável azimutal, definimos o conjunto de pontos desta quadratura a partir das equações (2.3) e (2.4) como:

$$\mu_{i,l} = \sin(\theta_i)\cos(\phi_l), \tag{4.113}$$

$$\eta_{i,l} = \sin(\theta_i)\sin(\phi_l) \tag{4.114}$$

e

$$W_{i,l} = w_i^p w_l^a, \tag{4.115}$$

com $i = 1, \ldots, N_\theta$, $l = 1, \ldots, N_\phi = N_{\phi_1} + N_{\phi_2}$, $W_{i,j}$ o peso da quadratura produto, w_i^p e w_l^a os pesos das quadraturas unidimensionais para as variáveis polar e azimutal, respectivamente.

Salientamos que os valores de N_{ϕ_1} e N_{ϕ_2} não necessitam serem iguais. Por exemplo, para $N_\theta = 2$, $N_{\phi_1} = 1$ e $N_{\phi_2} = 2$, temos uma quadratura com $M = 6$ direções discretas por octante. Em cada nível polar temos 3 direções azimutais, sendo uma no intervalo $[0, \phi_0)$ e as outras duas no intervalo $(\phi_0, \pi/2]$.

No capítulo seguinte, analisamos o desempenho numérico dos esquemas de quadratura apresentados neste trabalho.

5 RESULTADOS NUMÉRICOS

Neste capítulo, analisamos o desempenho em precisão dupla dos esquemas de quadratura para a esfera unitária apresentados no capítulo 4. Para tal, comparamos o erro relativo entre o valor analítico e o valor encontrado pela integração numérica da equação

$$\int_0^{\pi/2}\int_0^{\pi/2}(\sin\theta\cos\phi)^l(\sin\theta\sin\phi)^m\sin\theta d\phi d\theta = \sqrt{\pi}\frac{\Gamma(\frac{l+1}{2})\Gamma(\frac{m+1}{2})}{2\Gamma(\frac{l+m+3}{2})}, \tag{5.1}$$

cujo valor analítico é obtido através das equações (2.40) e (2.47). Consideramos a integração em um octante, dada pela equação (5.1), uma vez que, por exemplo, a quadratura QR é assim deduzida.

Também comparamos o erro relativo entre o valor analítico da equação (2.48) e o valor encontrado através dos esquemas de quadraturas. Estamos interessados na integração da equação (2.48) pois tem resultado analítico, que foi deduzido no capítulo 2, e tal integral considera integração em toda a esfera unitária. Uma vez que o valor analítico da equação (2.48) é zero sempre que l ou m seja um número inteiro ímpar, analisamos o desempenho destes esquemas de quadratura para a esfera unitária separando os casos em que l e m são pares dos casos em que um deles é ímpar.

Na seção 5.1 analisamos o desempenho dos esquemas de quadratura no octante principal da esfera unitária, e na seção 5.2 analisamos o desempenho na esfera unitária. Na seção 5.3 analisamos o desempenho dos esquemas de quadratura na integração da seguinte equação:

$$\int_0^{\pi}\int_0^{2\pi} f(\theta,\phi)\sin\theta d\phi d\theta = \pi^2 - \frac{32\sqrt{2}}{3}, \tag{5.2}$$

em que

$$f(\theta,\phi)=\begin{cases} f_1(\theta,\phi),\ \text{para } \theta\in(0,\pi),\phi\in[0,\pi/4]\cup[5\pi/4,2\pi) \\ f_2(\theta,\phi),\ \text{para } \theta\in(0,\pi),\phi\in(\pi/4,5\pi/4) \end{cases} \tag{5.3}$$

com

$$f_1(\theta,\phi)=\sin\theta+2\sin\theta[\sin\theta(\sin\phi-\cos\phi)-4\sin^2\theta\cos\phi\sin\phi], \tag{5.4}$$

$$f_2(\theta,\phi)=\sin\theta+2\sin\theta[\sin\theta(\cos\phi-\sin\phi)-4\sin^2\theta\cos\phi\sin\phi]. \tag{5.5}$$

Analisamos a integração numérica desta função, pois ao contrário das equações (2.48) e (5.1), esta função possui baixa regularidade, como será visto na seção 5.3. Para fins de cálculo, apresentamos na tabela 5.1 o número de direções discretas por octante para cada esquema de quadratura de ordem N (para as quadraturas P_NT_N e $P_NT_NS_N$) ou N_θ (para as quadraturas QR).

Tabela 5.1 Esquema de quadratura de ordem N ou N_θ e respectivo número de direções discretas por octante

Quadratura	M
P_NT_N	$N^2/4$
$P_NT_NS_N$	$N(N+2)/8$
QR_QN_θ	N_θ^2
$QR\_T_{N_\theta}$	$N_\theta(N_\theta+1)/2$

Nota: $QR\_Q_{N_\theta}$ e $QR\_T_{N_\theta}$ representam todas as quadraturas QR com acoplamentos quadrangular e triangular, respectivamente

5.1 Análise para o Octante Principal

Nesta seção, analisamos o desempenho numérico dos esquemas de quadratura para a esfera unitária apresentados no capítulo 4 através da equação (5.1), para $l, m = 0, \ldots, 50$. Tais esquemas de quadraturas são gerados numericamente em Fortran 95 e resolvemos os problemas de autovalores necessários através da rotina *iter_qr*. Para os esquemas de quadratura QR, utilizamos os parâmetros $h = 0.004$ e $tol = \varepsilon_M$, e geramos as quadraturas associadas as variáveis polar e azimutal através da rotina *fn_qrs_full_tanh_sinh*.

5.1.1 Quadratura $P_N T_N$

O esquema de quadratura $P_N T_N$ demonstrou um bom desempenho numérico quando consideramos que l e m são pares ou quando $l \geq 10$ e $m \geq 10$. Para estes valores de l e m, utilizando a quadratura de ordem $N = 64$, observamos que o maior erro relativo encontrado foi de ordem $\mathcal{O}(10^{-11})$, sendo que somente 177 (de 1681) valores obtiveram erros relativos com ordem superior à $\mathcal{O}(10^{-15})$. Contudo, quando consideramos valores de l ou m ímpares e menores que 10, encontramos erros relativos de ordem $\mathcal{O}(10^{-3})$.

5.1.2 Quadratura $P_N T_N S_N$

De modo semelhante à quadratura $P_N T_N$, o esquema de quadratura $P_N T_N S_N$ tem desempenho satisfatório quando consideramos valores para l e m pares ou $l \geq 10$ e $m \geq 10$. Para tais valores de l e m, utilizando a ordem de quadratura $N = 64$, todos os erros relativos obtidos são de ordem $\mathcal{O}(10^{-10})$ ou menores, mas

238 (de 1681) valores obtiveram erros relativos com ordem superior à $\mathcal{O}(10^{-15})$. Considerando valores de l ou m ímpares e menores que 10, erros relativos de ordem $\mathcal{O}(10^{-3})$ já começam a aparecer.

5.1.3 Quadratura QR Quadrangular

Ao contrário do que acontece com as quadraturas P_NT_N e $P_NT_NS_N$, o esquema de quadratura $QRS45m\_Q_{N_\theta}$ integra bem a equação (5.1) para todos os valores de $l, m = 0, \ldots, 50$, e com isso, para valores de $l, m \leq 10$, esta quadratura é mais adequada que as anteriores. Para a ordem $N_\theta = 32$, todos os erros relativos são de ordem $\mathcal{O}(10^{-10})$ ou menores, sendo que, os casos em que $l, m \geq 10$, observamos que somente 19 (de 1681) valores possuem erro relativo de ordem $\mathcal{O}(10^{-15})$ ou menores, de modo que, para estes valores de l e m, esta quadratura é levemente inferior às quadraturas P_NT_N e $P_NT_NS_N$. Em particular, para este caso, 1425 (de 1681) valores possuem erro relativo de ordem $\mathcal{O}(10^{-14})$. Por outro lado, quando analisamos somente os casos em que $l, m < 10$, observamos que todos os erros relativos são de ordem $\mathcal{O}(10^{-14})$.

Semelhante a quadratura $QRS45m\_Q_{N_\theta}$, o esquema de quadratura $QRA45m\_Q_{N_\theta}$ tem um bom desempenho numérico ao integrar a equação (5.1) para todos os valores de $l, m = 0, \ldots, 50$, tornando-se uma melhor opção para a integração de potências com $l, m \leq 10$, em comparação às quadraturas P_NT_N e $P_NT_NS_N$. Contudo, seu desempenho é levemente inferior ao do esquema de quadratura $QRS45m\_Q_{N_\theta}$, pois utilizando a ordem $N_\theta = 32$, apresenta erros relativos de ordem $\mathcal{O}(10^{-8})$ ou menores, e analisando somente os casos em que $l, m \geq 10$, observamos 1060 (de 1681) valores com erro relativo de ordem $\mathcal{O}(10^{-14})$ ou menor. Para

os casos em que $l, m < 10$, o desempenho dessa quadratura é análogo ao esquema $QRS45m\_Q_{N_\theta}$, apresentando todos os erros relativos de ordem $\mathcal{O}(10^{-14})$.

Já o esquema de quadratura $QRJ45m\_Q_{N_\theta}$ tem um bom desempenho numérico para valores de l e m pares ou quando $l \geq 10$ e $m \geq 10$. Analisando somente os valores pares de l e m e utilizando a quadratura de ordem $N_\theta = 32$, todos os erros relativos são de ordem $\mathcal{O}(10^{-14})$; considerando somente $l \geq 10$ e $m \geq 10$, observamos que todos os erros relativos são de ordem $\mathcal{O}(10^{-11})$, sendo que 1565 (de 1681) valores possuem erro relativo de ordem $\mathcal{O}(10^{-14})$ ou menor. Contudo, para valores de $l, m < 10$, erros relativos de ordem $\mathcal{O}(10^{-3})$ começam a surgir.

O esquema de quadratura $QRS90m\_Q_{N_\theta}$, que se diferencia das quadraturas QR anteriores por não possuir simetria em relação a $\phi = \pi/4$, quando analisando os casos em que $l, m \geq 10$, teve um desempenho numérico superior ao esquema de quadratura $QRA45m\_Q_{N_\theta}$ e inferior aos demais, visto que, utilizando a ordem de quadratura $N_\theta = 32$, observamos erros relativos de ordem $\mathcal{O}(10^{-9})$ e o total de 1462 (de 1681) valores com erro relativo de ordem $\mathcal{O}(10^{-14})$ ou menor. Além disso, nos casos em que $l \geq 10$ e $m \leq 5$, este esquema teve desempenho satisfatório, apresentando erros relativos de ordem $\mathcal{O}(10^{-14})$ ou inferior. Contudo, para valores em pequenos de l ($l < 10$), o esquema se mostrou ineficaz, obtendo erros relativos a partir da ordem $\mathcal{O}(10^{-3})$.

Por fim, o esquema de quadratura $QRJ90m\_Q_{N_\theta}$, que também não possui simetria em relação a $\phi = \pi/4$, mostrou-se capaz de integrar de forma satisfatória a equação (5.1) para todos os valores de $l, m = 0, \ldots, 50$. Para valores de $l, m \geq 10$, utilizando a ordem de quadratura $N_\theta = 32$, observamos erros relativos de ordem $\mathcal{O}(10^{-8})$, apresentando o total de 1050 (de 1681) valores com erro relativo de

ordem $\mathcal{O}(10^{-14})$ ou menor, tornando o desempenho desta quadratura semelhante ao desempenho da quadratura $QRA45m\_Q_{N_\theta}$. Já para os valores de $l, m < 10$, para $N_\theta = 32$, observamos erros relativos todos da ordem $\mathcal{O}(10^{-14})$, demonstrando um bom desempenho numérico.

5.1.4 Quadratura QR Triangular

O esquema de quadratura $QRS45m\_T_{N_\theta}$ demonstrou um bom desempenho numérico ao integrar a equação (5.1) para todos os valores de $l, m = 0, \ldots, 50$. Para a ordem $N_\theta = 32$, todos os erros relativos são de ordem $\mathcal{O}(10^{-8})$ ou menores, e quando analisamos os casos em que $l, m \geq 10$, observamos que todos os erros são de ordem $\mathcal{O}(10^{-11})$ ou menor, sendo que 1554 (de 1681) dos valores possuem erro relativo de ordem $\mathcal{O}(10^{-14})$ ou menor.

Por sua vez, o esquema de quadratura $QRA45m\_T_{N_\theta}$ também é capaz de integrar a equação (5.1) para todos os valores de $l, m = 0, \ldots, 50$, com um desempenho levemente inferior ao esquema $QRS45m\_T_{N_\theta}$. Para a ordem $N_\theta = 32$, todos os erros relativos são de ordem inferior a $\mathcal{O}(10^{-8})$, e quando analisamos os casos em que $l, m \geq 10$, observamos que surgem erros relativos a partir da ordem $\mathcal{O}(10^{-9})$, com o total de 1217 (de 1681) valores com erro relativo de ordem 10^{-14} ou inferior.

Já a quadratura $QRJ45m\_T_{N_\theta}$ demonstrou um bom desempenho numérico ao integrar a equação (5.1) para valores de l e m pares ou para $l, m \geq 10$, mas não foi capaz de integrar com boa precisão valores de $l, m < 10$. Para a ordem de quadratura $N_\theta = 32$, analisando somente os valores de l e m pares, todos os erros relativos são da ordem $\mathcal{O}(10^{-14})$; quando consideramos o caso $l, m \geq 10$, surgem erros relativos a partir da ordem $\mathcal{O}(10^{-11})$, sendo que 1536 (de 1681) valores possuem

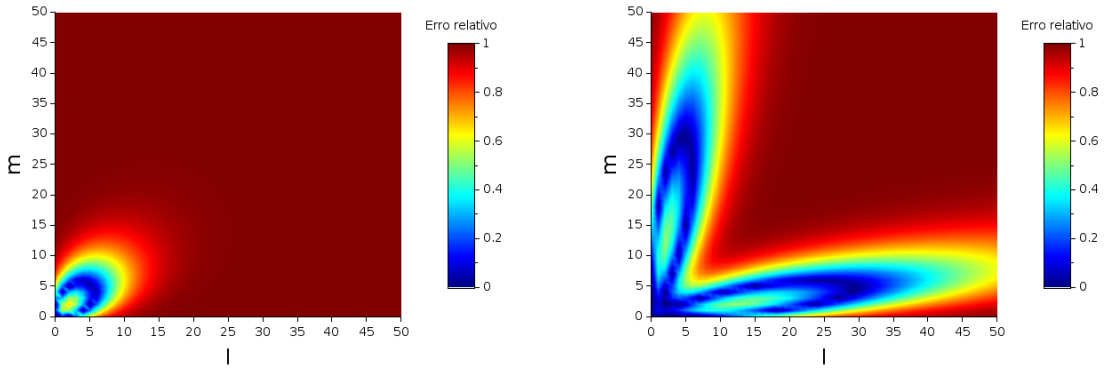
erro relativo de ordem $\mathcal{O}(10^{-14})$ ou menor. Contudo, para $l, m < 10$, encontramos valores com erro relativo da ordem $\mathcal{O}(10^{-3})$.

O esquema de quadratura $QRS90m\_T_{N_\theta}$, quando analisamos os casos em que $l, m \geq 10$, apresentou um desempenho numérico satisfatório, e assim como a quadratura $QRS90m\_Q_{N_\theta}$, obteve bom desempenho para os casos em que $l \geq 10$ e $m \leq 5$. Para a ordem $N_\theta = 32$, nos casos em que $l, m \geq 10$, observamos erros relativos a partir da ordem $\mathcal{O}(10^{-9})$, sendo que 1305 (de 1681) valores possuem erro relativo de ordem $\mathcal{O}(10^{-14})$ ou menor; nos casos em que $l \geq 10$ e $m \leq 5$, observamos dois valores com erro relativo de ordem $\mathcal{O}(10^{-13})$ e os demais com ordem de $\mathcal{O}(10^{-14})$. Contudo, para valores pequenos de l ($l < 10$), a quadratura se mostrou ineficaz, obtendo erros relativos a partir da ordem $\mathcal{O}(10^{-3})$.

Por fim, o esquema de quadratura $QRJ90m\_T_{N_\theta}$ foi capaz de integrar de forma satisfatório a equação (5.1) para todos os valores de $l, m = 0, \ldots, 50$. Para a ordem de quadratura $N_\theta = 32$, analisando os casos em que $l, m \geq 10$, observamos erros relativos a partir da ordem $\mathcal{O}(10^{-9})$, sendo que 1231 (de 1681) valores possuem erro relativo de ordem $\mathcal{O}(10^{-14})$ ou menor. Já para os casos em quem $l, m < 10$, observamos erros relativos com ordem $\mathcal{O}(10^{-8})$ ou menor.

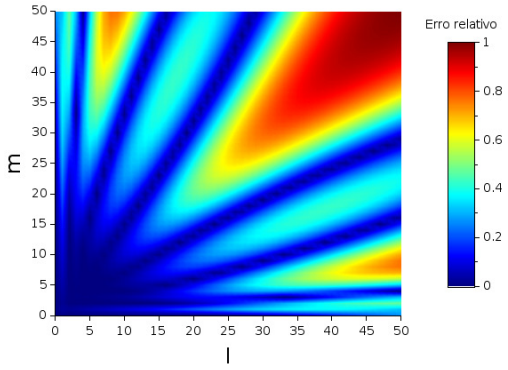
Nas figuras 5.1−5.12, com o intuito de mostrar que o aumento na ordem da quadratura diminui os erros relativos, apresentamos o erro relativo (normalizado para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura numérica de algumas ordens das quadraturas discutidas nas seções 5.1.1−5.1.4, para $l, m = 0, \ldots, 50$. No apêndice B, apresentamos os valores mínimo e máximo do erro relativo entre o valor analítico e o da quadratura numérica (no octante principal) para algumas ordens de

quadratura dos esquemas abordados nesta seção. Na próxima seção, analisamos o desempenho destes esquemas de quadratura para a esfera unitária.

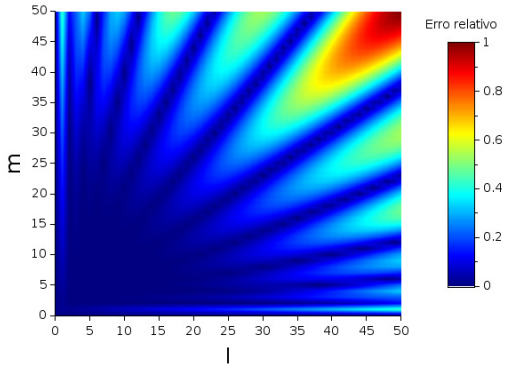


(a) $N = 2, M = 1$

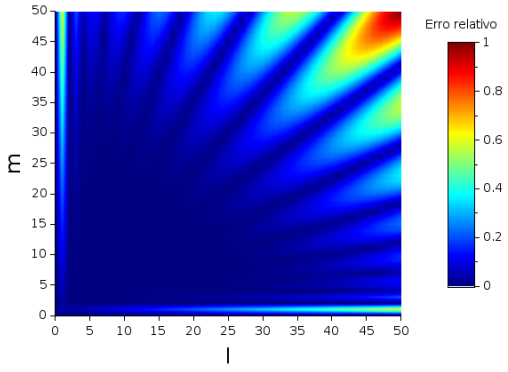
(b) $N = 4, M = 4$



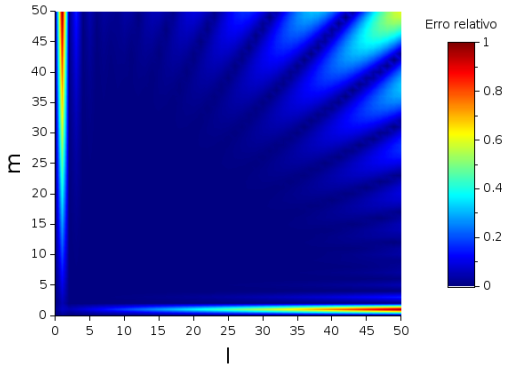
(c) $N = 8, M = 16$



(d) $N = 12, M = 36$

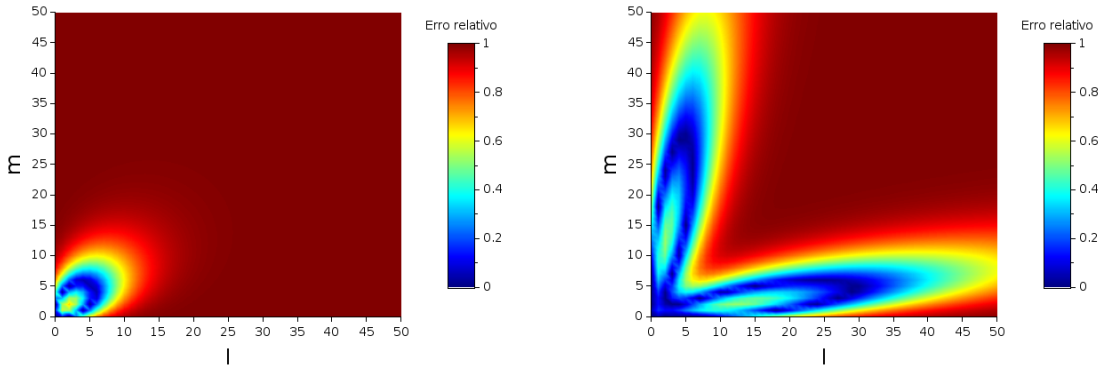


(e) $N = 16, M = 64$

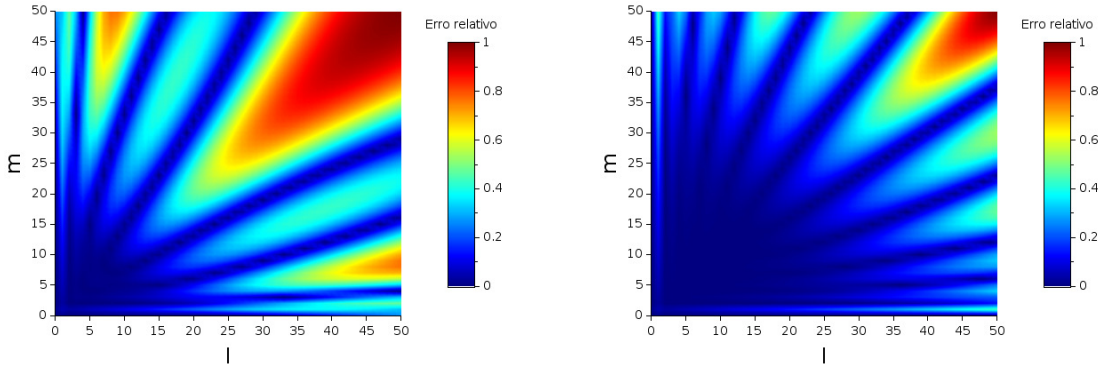


(f) $N = 20, M = 100$

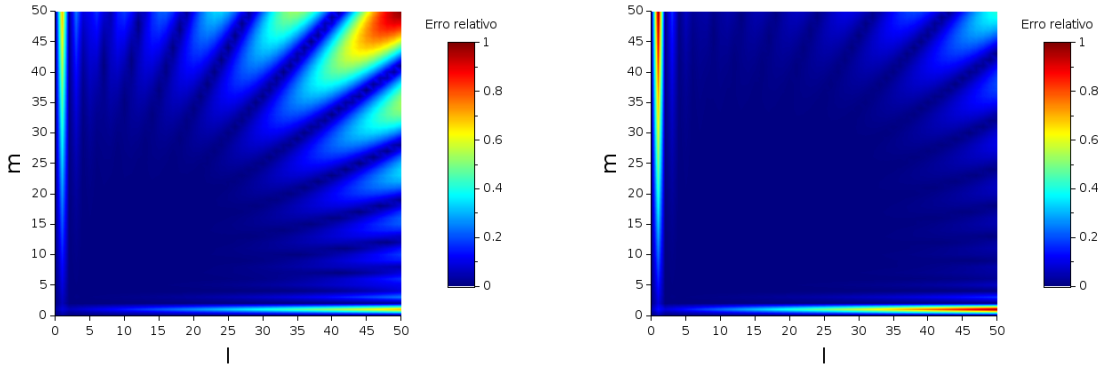
Figura 5.1 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $P_N T_N$



(a) $N = 2, M = 1$ (b) $N = 4, M = 3$

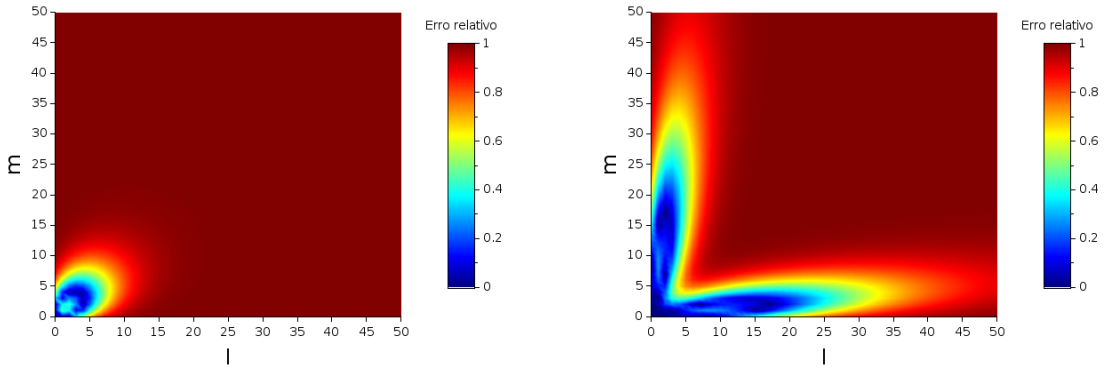


(c) $N = 8, M = 10$ (d) $N = 12, M = 21$



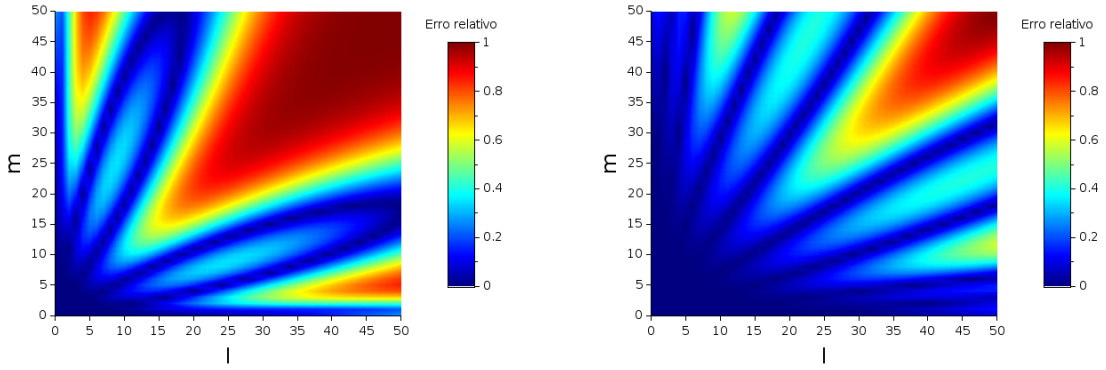
(e) $N = 16, M = 36$ (f) $N = 20, M = 55$

Figura 5.2 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $P_N T_N S_N$



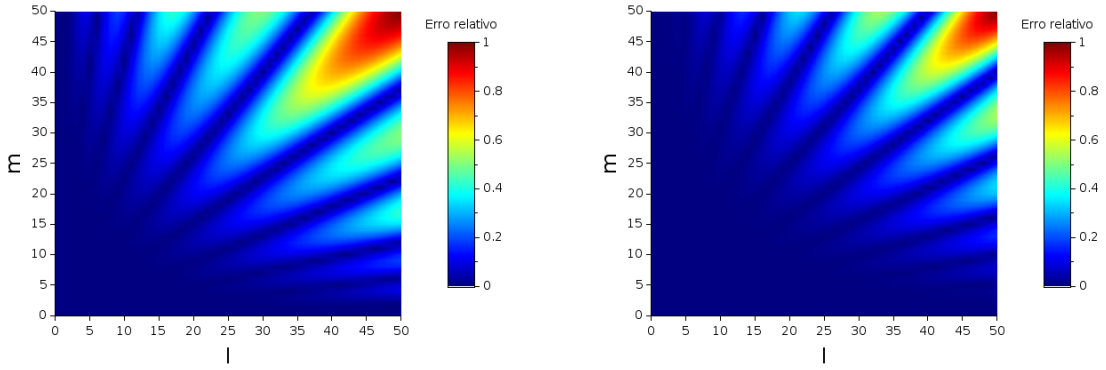
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 4$



(c) $N_\theta = 4, M = 16$

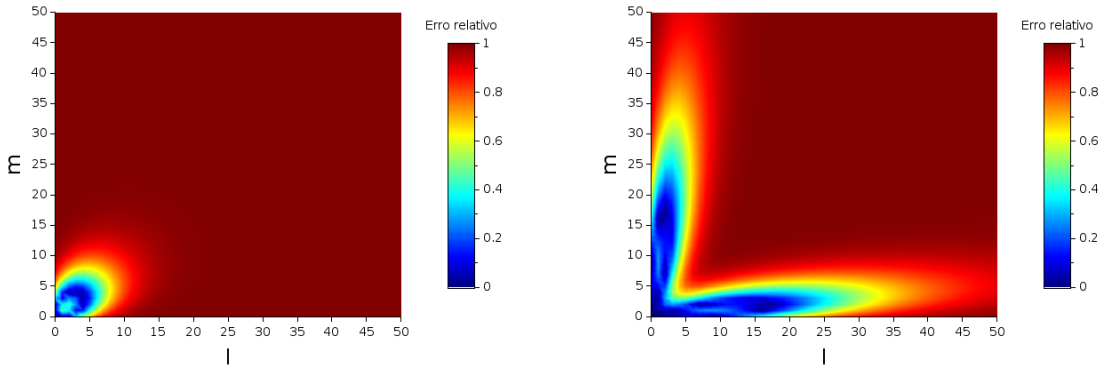
(d) $N_\theta = 6, M = 36$



(e) $N_\theta = 8, M = 64$

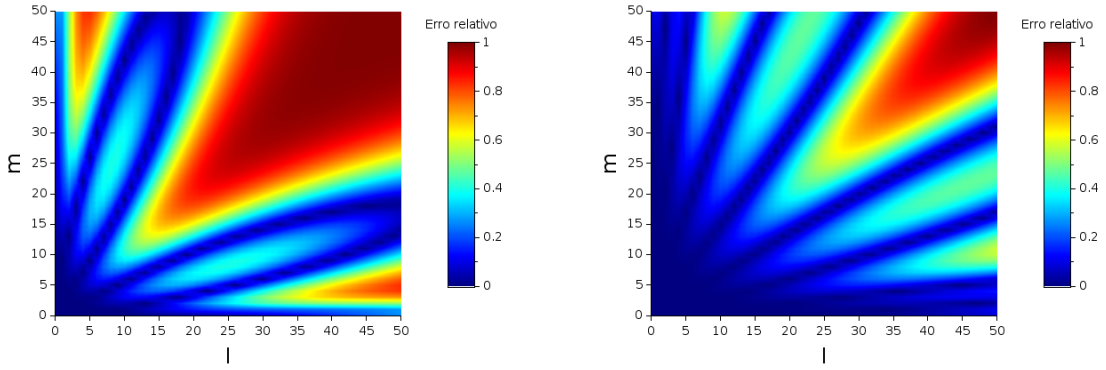
(f) $N_\theta = 10, M = 100$

Figura 5.3 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRS45m\_Q_{N_\theta}$



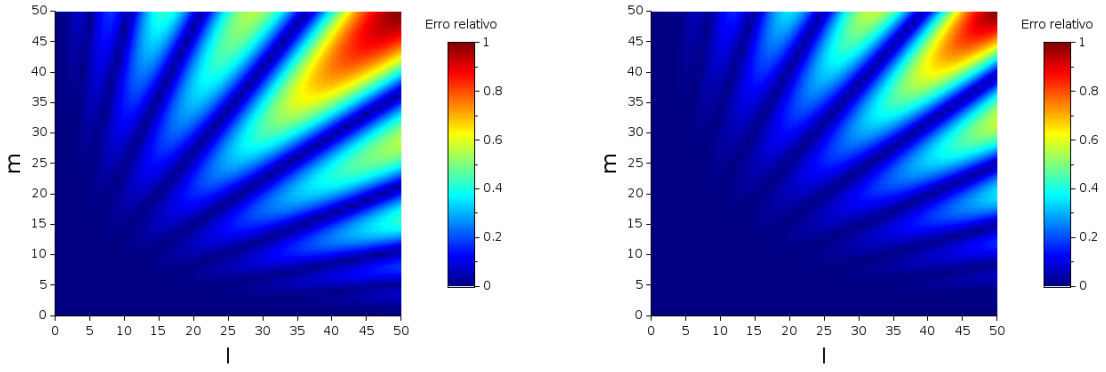
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 4$



(c) $N_\theta = 4, M = 16$

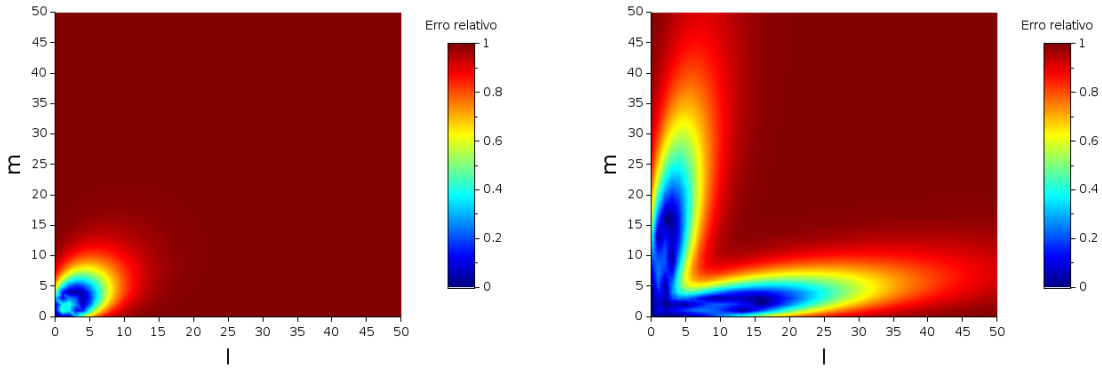
(d) $N_\theta = 6, M = 36$



(e) $N_\theta = 8, M = 64$

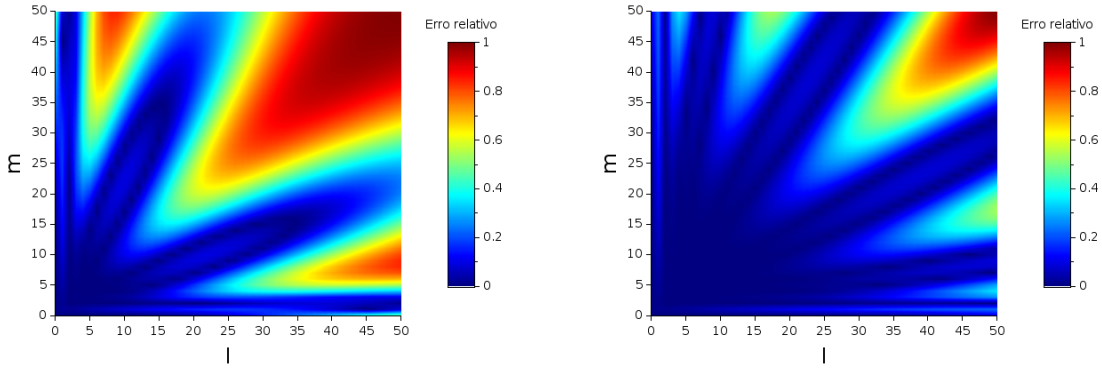
(f) $N_\theta = 10, M = 100$

Figura 5.4 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRA45m\_Q_{N_\theta}$



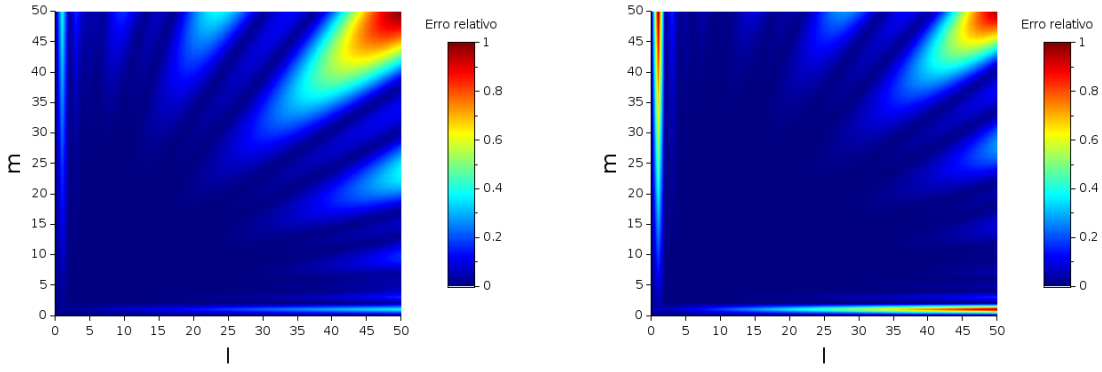
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 4$



(c) $N_\theta = 4, M = 16$

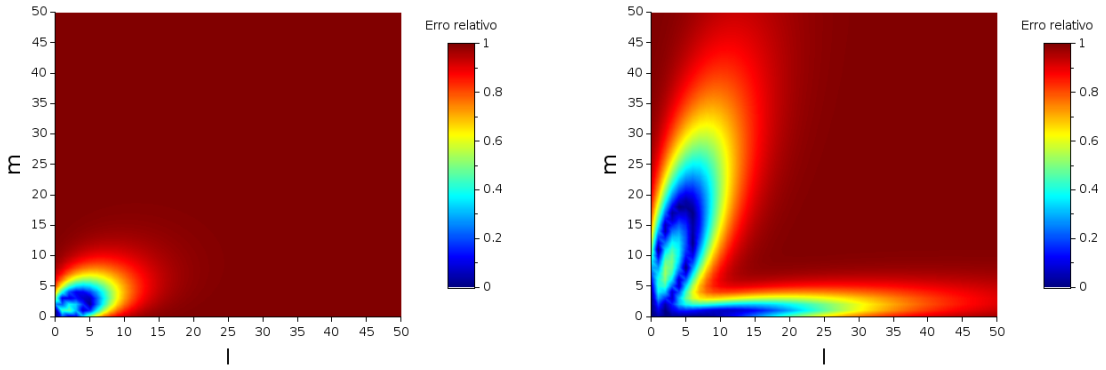
(d) $N_\theta = 6, M = 36$



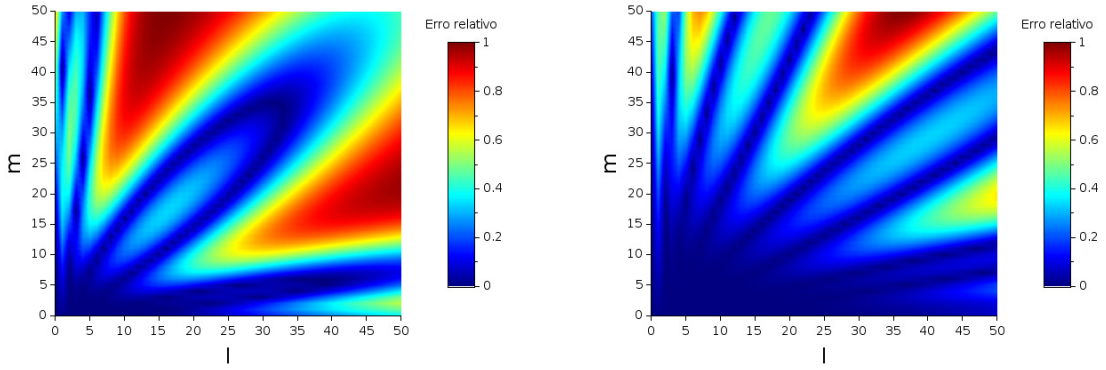
(e) $N_\theta = 8, M = 64$

(f) $N_\theta = 10, M = 100$

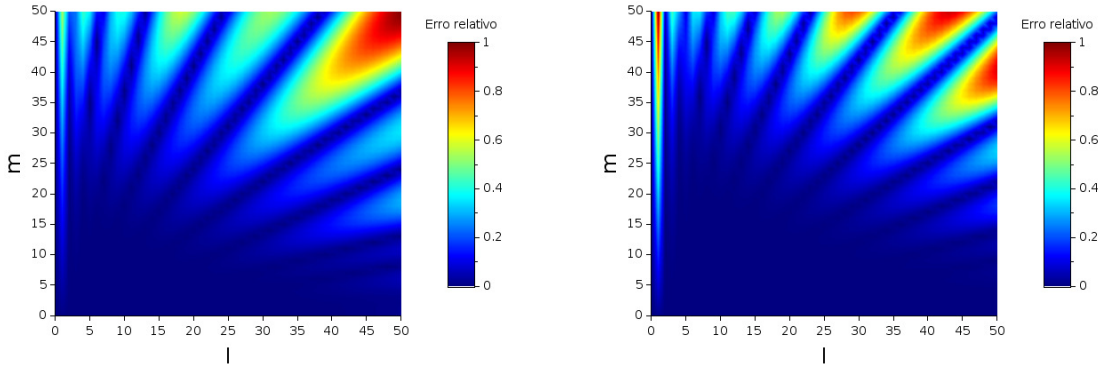
Figura 5.5 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRJ45m\_Q_{N_\theta}$



(a) $N_\theta = 1, M = 1$ (b) $N_\theta = 2, M = 4$

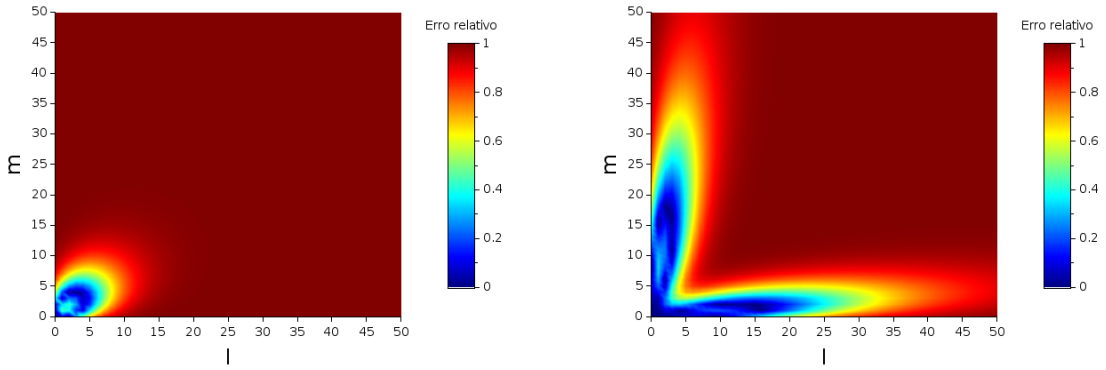


(c) $N_\theta = 4, M = 16$ (d) $N_\theta = 6, M = 36$

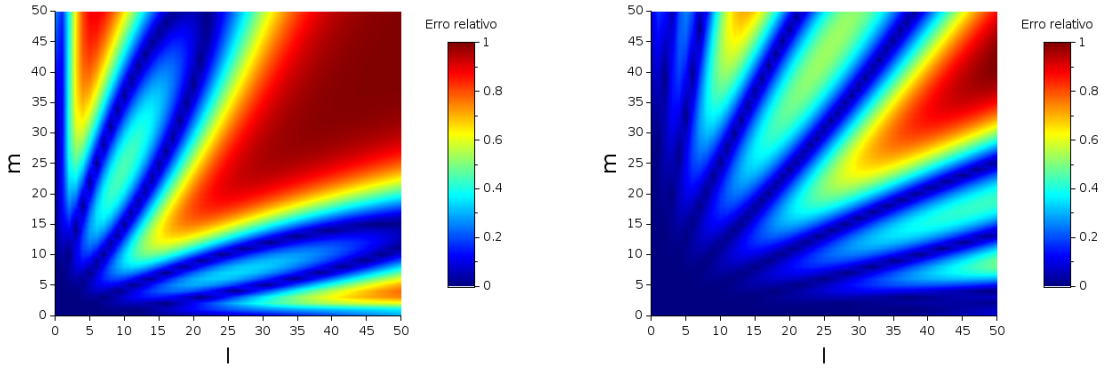


(e) $N_\theta = 8, M = 64$ (f) $N_\theta = 10, M = 100$

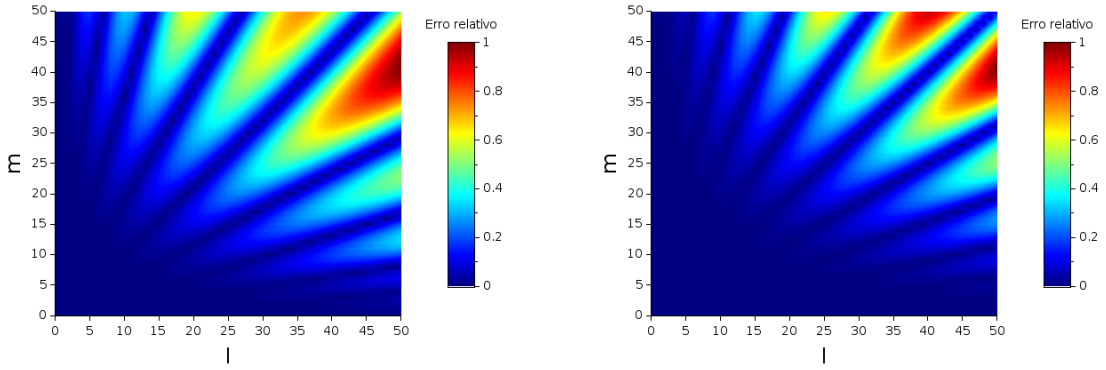
Figura 5.6 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRS90m\_Q_{N_\theta}$



(a) $N_\theta = 1, M = 1$ (b) $N_\theta = 2, M = 4$

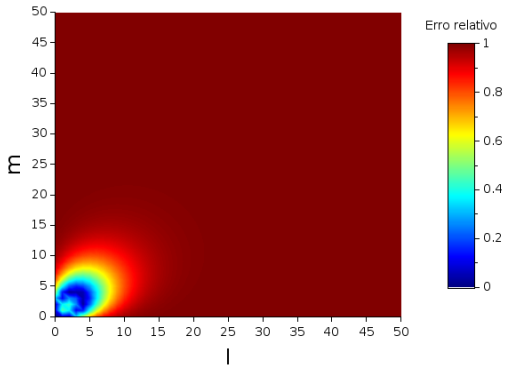


(c) $N_\theta = 4, M = 16$ (d) $N_\theta = 6, M = 36$

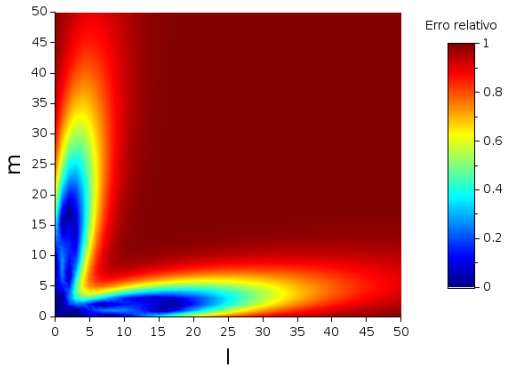


(e) $N_\theta = 8, M = 64$ (f) $N_\theta = 10, M = 100$

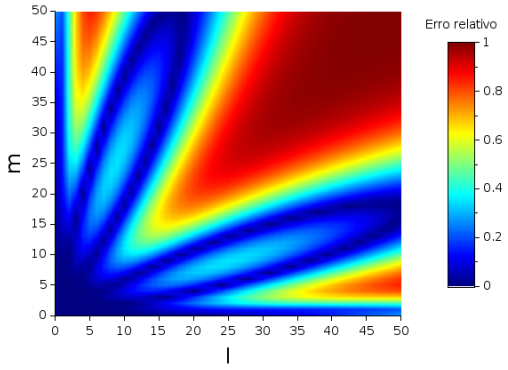
Figura 5.7 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRJ90m\_Q_{N_\theta}$



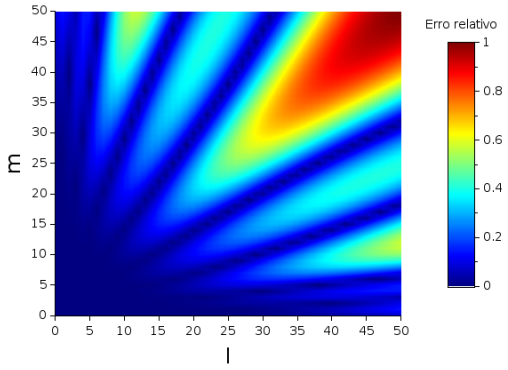
(a) $N_\theta = 1, M = 1$



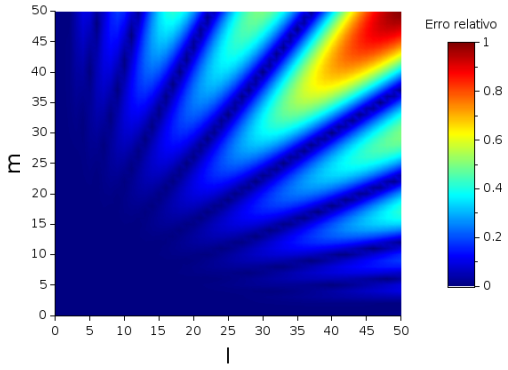
(b) $N_\theta = 2, M = 3$



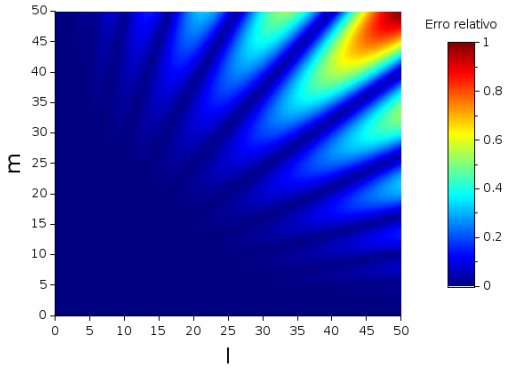
(c) $N_\theta = 4, M = 10$



(d) $N_\theta = 6, M = 21$

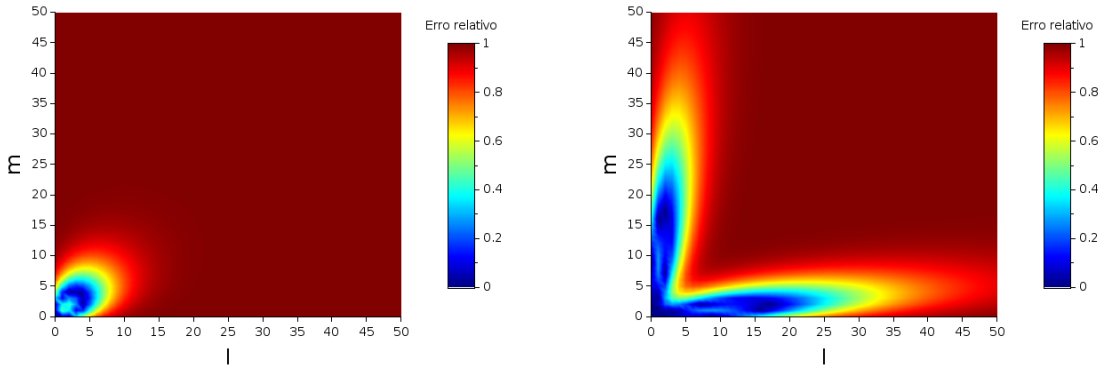


(e) $N_\theta = 8, M = 36$



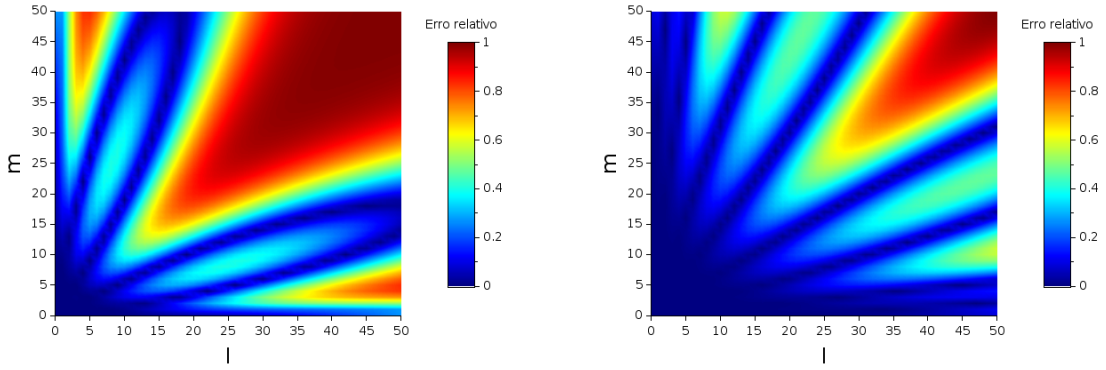
(f) $N_\theta = 10, M = 55$

Figura 5.8 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRS45m\_T_{N_\theta}$



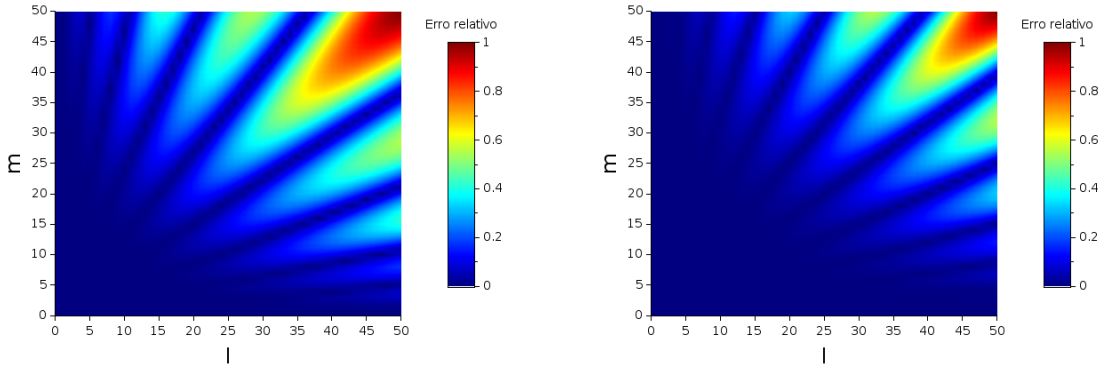
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 3$



(c) $N_\theta = 4, M = 10$

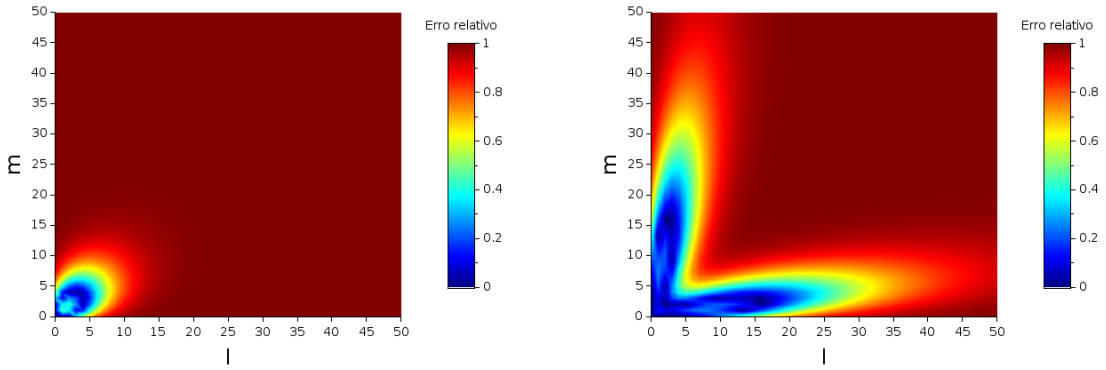
(d) $N_\theta = 6, M = 21$



(e) $N_\theta = 8, M = 36$

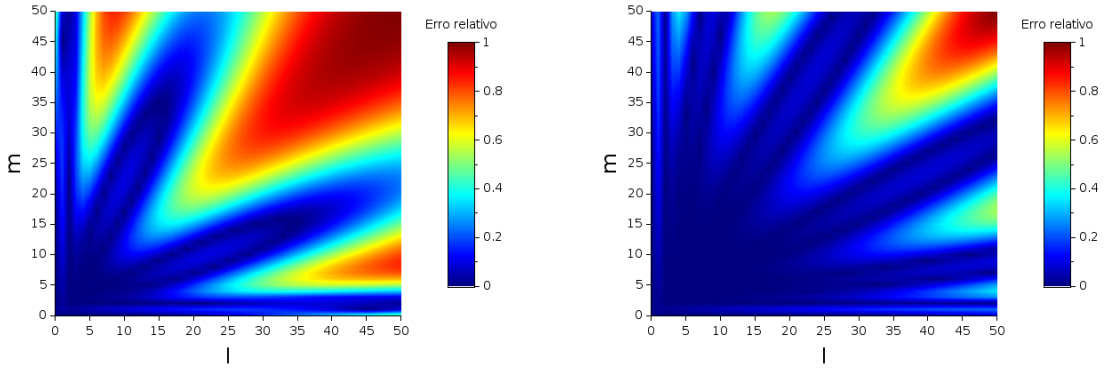
(f) $N_\theta = 10, M = 55$

Figura 5.9 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRA45m\_T_{N_\theta}$



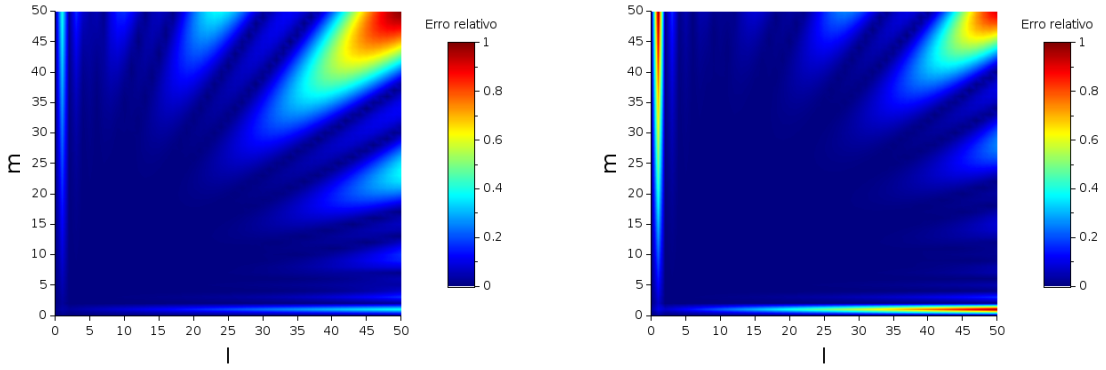
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 3$



(c) $N_\theta = 4, M = 10$

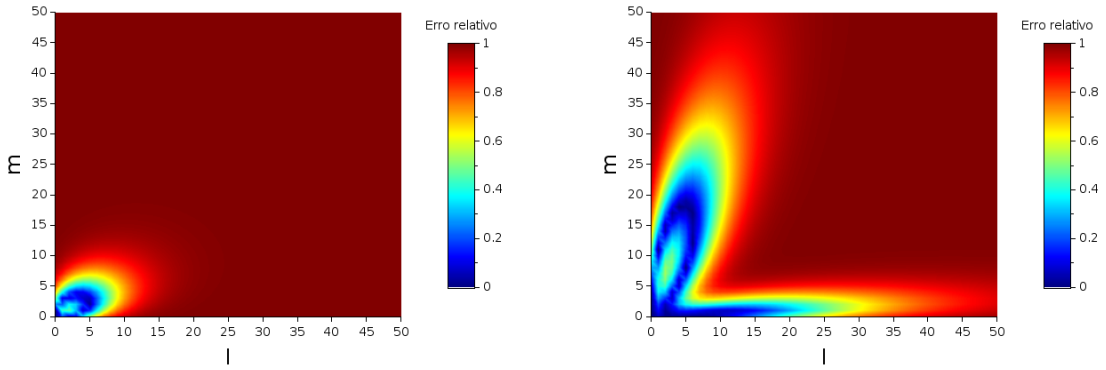
(d) $N_\theta = 6, M = 21$



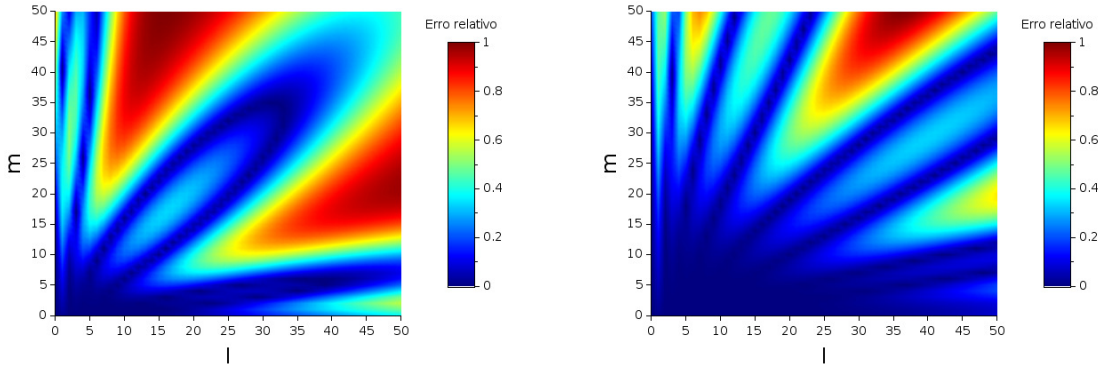
(e) $N_\theta = 8, M = 36$

(f) $N_\theta = 10, M = 55$

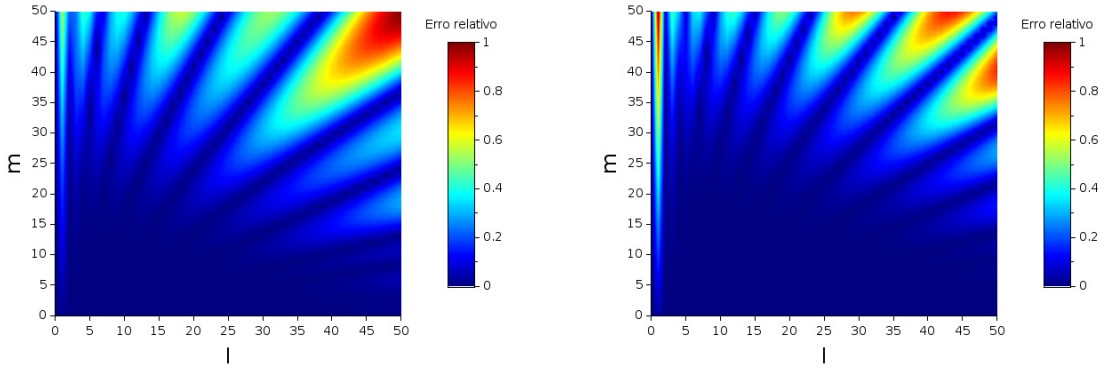
Figura 5.10 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRJ45m\_T_{N_\theta}$



(a) $N_\theta = 1, M = 1$ (b) $N_\theta = 2, M = 3$

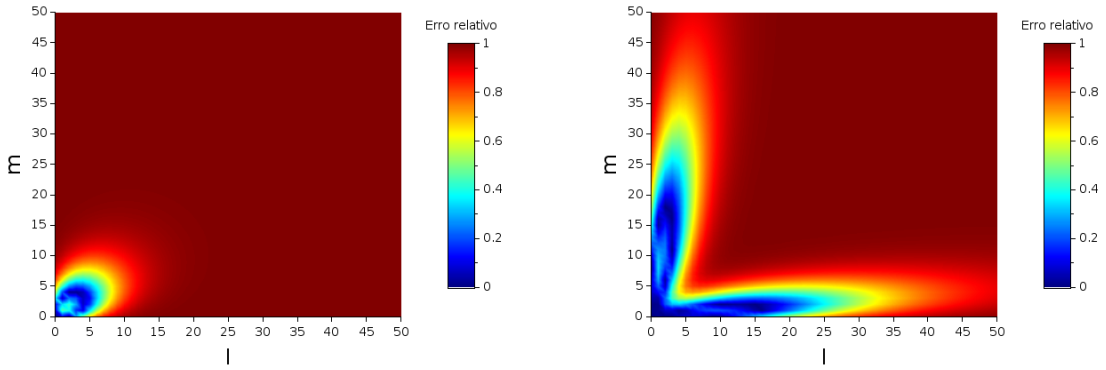


(c) $N_\theta = 4, M = 10$ (d) $N_\theta = 6, M = 21$

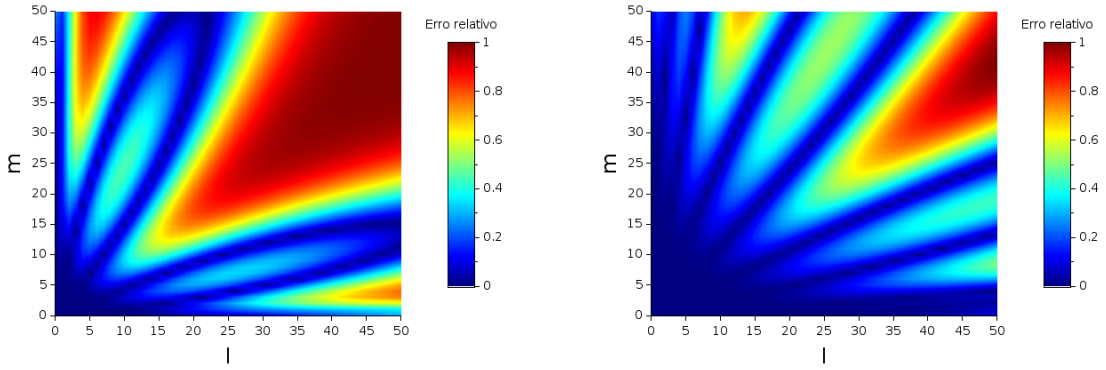


(e) $N_\theta = 8, M = 36$ (f) $N_\theta = 10, M = 55$

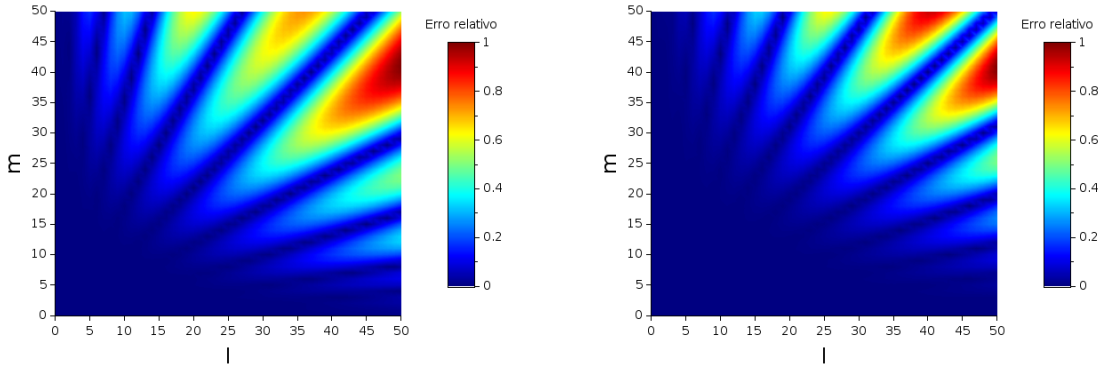
Figura 5.11 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRS90m\_T_{N_\theta}$



(a) $N_\theta = 1, M = 1$ (b) $N_\theta = 2, M = 3$



(c) $N_\theta = 4, M = 10$ (d) $N_\theta = 6, M = 21$



(e) $N_\theta = 8, M = 36$ (f) $N_\theta = 10, M = 55$

Figura 5.12 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRJ90m\_T_{N_\theta}$

5.2 Análise para a Esfera Unitária

Nesta seção, analisamos o desempenho numérico dos esquemas de quadratura estudados no capítulo 4 através da equação (2.48), que envolve uma integração na esfera unitária, para $l, m = 0, \ldots, 50$. Em nossa análise, consideramos as ordens $N = 64$ (para as quadraturas $P_N T_N$ e $P_N T_N S_N$) e $N_\theta = 32$ (para as quadraturas QR), e consideramos dois casos para cada esquema de quadratura: o caso em que l e m são pares e o caso em que l ou m é ímpar. Para o caso de l e m pares, analisamos o erro relativo entre o valor analítico e o encontrado pela quadratura; para o caso de l ou m ser ímpar, analisamos o erro absoluto, uma vez que neste caso o valor analítico é zero.

Assim como na seção anterior, os esquemas de quadraturas são gerados numericamente em Fortran 95 e resolvemos os problemas de autovalores necessários através da rotina *iter_ qr*. Para os esquemas de quadratura QR, utilizamos os parâmetros $h = 0.004$ e $tol = \varepsilon_M$, e geramos as quadraturas associadas as variáveis polar e azimutal através da rotina *fn_ qrs_full_ tanh_ sinh*.

5.2.1 Quadratura $P_N T_N$

Primeiramente, analisamos os casos em que l e m são inteiros pares e observamos que somente um caso obteve erro relativo de ordem $\mathcal{O}(10^{-14})$, enquanto os demais todos são de ordem inferior. Para os casos em que l ou m é ímpar, observamos que o maior erro absoluto encontrado é de ordem $\mathcal{O}(10^{-14})$, o que significa que para estes casos o esquema de quadratura também aproxima bem a função.

5.2.2 Quadratura $P_N T_N S_N$

Nos casos em que l e m são pares, esta quadratura obteve desempenho inferior ao da quadratura $P_N T_N$, pois apresenta erros relativos a partir da ordem $\mathcal{O}(10^{-9})$. Já para os casos em que l ou m é impar, observamos que o maior erro absoluto encontrado é de ordem $\mathcal{O}(10^{-14})$, de modo que este esquema integra bem a função para estes casos.

5.2.3 Quadratura QR Quadrangular

Nos casos em que l e m são pares, o esquema de quadratura $QRS45m\_Q_{N_\theta}$ mostrou desempenho semelhante ao obtido para o octante principal, pois encontramos erros relativos de ordem $\mathcal{O}(10^{-10})$ ou menor, sendo que 602 (de 676) valores possuem erro relativo de ordem $\mathcal{O}(10^{-14})$. Por sua vez, os casos em que l ou m é ímpar, o maior erro absoluto encontrado foi de ordem $\mathcal{O}(10^{-14})$, mostrando que este esquema é adequado para estes casos.

O esquema de quadratura $QRA45m\_Q_{N_\theta}$ também obteve desempenho semelhante ao obtido para o octante principal, obtendo todos os erros relativos de ordem inferior à $\mathcal{O}(10^{-8})$, sendo que 512 (de 676) valores obtiveram erros de ordem $\mathcal{O}(10^{-14})$ ou menor. Para os casos em que l ou m é impar, o esquema demonstrou desempenho satisfatório, já que o maior erro absoluto encontrado foi de ordem $\mathcal{O}(10^{-14})$.

Já o esquema de quadratura $QRJ45m\_Q_{N_\theta}$ obteve o melhor desempenho dentre todas as quadraturas QR, pois para os casos em que l e m são pares,

todos os erros relativos são de ordem $\mathcal{O}(10^{-14})$, e para os casos em que l ou m é ímpar, o maior erro absoluto também é de ordem $\mathcal{O}(10^{-14})$.

O esquema de quadratura $QRS90m\_Q_{N_\theta}$ obteve o segundo melhor desempenho dentre as quadraturas QR ao integrar a equação (2.48) para valores de l e m pares, visto que apresenta erros relativos de ordem $\mathcal{O}(10^{-12})$ ou menor, com 639 (de 576) valores com erro relativo de ordem $\mathcal{O}(10^{-14})$ ou menor. Além disso, para os casos em que l ou m é ímpar, o maior erro absoluto encontrado foi de ordem $\mathcal{O}(10^{-14})$, demonstrando desempenho equivalente as demais quadraturas para este caso.

Por fim, o esquema de quadratura $QRJ90m\_Q_{N_\theta}$ obteve desempenho semelhante ao do esquema $QRA45m\_Q_{N_\theta}$, pois o maior erro absoluto é na ordem de $\mathcal{O}(10^{-14})$ para os casos em que l ou m é ímpar, e apresentando erros relativos a partir da ordem $\mathcal{O}(10^{-8})$ para os casos em que l e m são pares.

5.2.4 Quadratura QR Triangular

Analisando a quadratura $QRS45m\_T_{N_\theta}$ nos casos em que l e m são pares, observamos erros relativos de ordem $\mathcal{O}(10^{-11})$ ou menor, sendo que 633 (de 676) valores possuem ordem de $\mathcal{O}(10^{-14})$. Já para os casos em que l ou m é ímpar, o maior erro absoluto encontrado é da ordem $\mathcal{O}(10^{-14})$, assim como as quadraturas QR anteriores.

O desempenho obtido pelo esquema de quadratura $QRA45m\_T_{N_\theta}$ para a esfera unitária foi semelhante ao obtido para o octante principal. Para os casos em que l e m são pares, observamos erros relativos de ordem $\mathcal{O}(10^{-9})$ ou menor, sendo

que 555 (de 676) valores possuem erros relativos de ordem $\mathcal{O}(10^{-14})$ ou menor. Já para os casos em que l ou m é ímpar, o maior erro absoluto observado é na ordem de $\mathcal{O}(10^{-15})$.

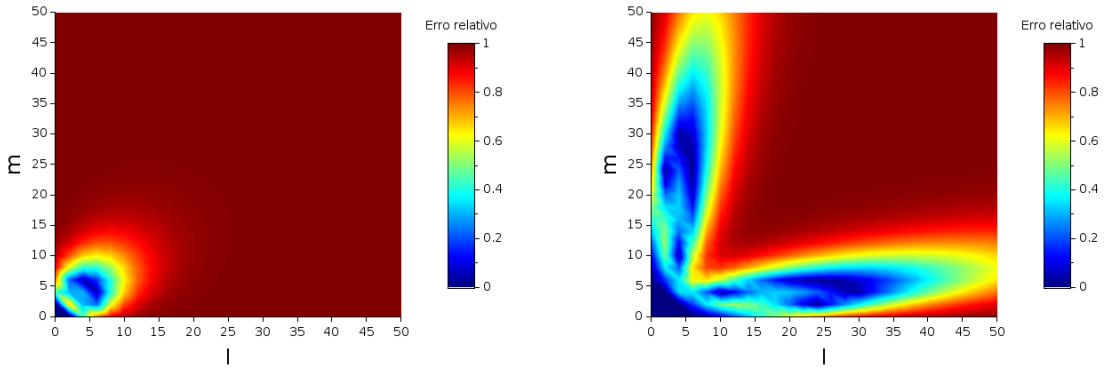
O esquema de quadratura $QRJ45m\_T_{N_\theta}$ demonstrou o melhor desempenho dentre as quadraturas QR triangulares para os casos em que l e m são pares, obtendo todos os erros relativos da ordem $\mathcal{O}(10^{-14})$, e obtendo um desempenho equivalente aos demais no caso em que l ou m é ímpar, uma vez que o maior erro absoluto encontrado é da ordem $\mathcal{O}(10^{-14})$.

A quadratura $QRS90m\_T_{N_\theta}$ obteve desempenho semelhante ao da quadratura $QRS45m\_T_{N_\theta}$. Nos casos em que l e m são pares, observamos erros relativos a partir da ordem $\mathcal{O}(10^{-11})$, sendo que 605 (de 676) valores obtiveram erros relativos de ordem $\mathcal{O}(10^{-14})$ ou inferior. Já nos casos em que l ou m é ímpar, o desempenho é semelhante as demais quadraturas, em que observamos que o maior erro absoluto na ordem de $\mathcal{O}(10^{-14})$.

Por fim, o esquema de quadratura $QRJ90m\_T_{N_\theta}$ obteve desempenho semelhante ao da quadratura $QRA45m\_T_{N_\theta}$. Nos casos em que l e m são pares, observamos erros relativos a partir da ordem $\mathcal{O}(10^{-9})$, e 556 (de 676) valores obtiveram erros relativos de ordem $\mathcal{O}(10^{-14})$ ou inferior. Nos casos em que l ou m é ímpar, o maior erro absoluto observado é ordem de $\mathcal{O}(10^{-14})$.

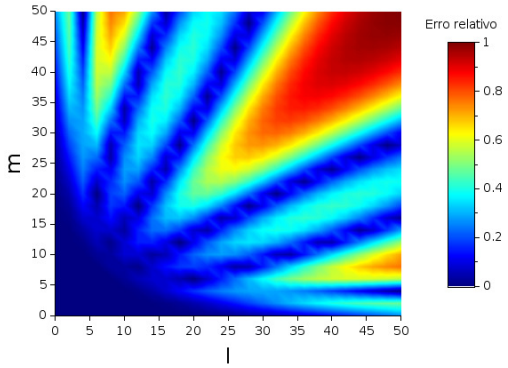
Nas figuras 5.13−5.24, com o intuito de mostrar que o aumento na ordem da quadratura diminui os erros relativos nos casos em que l e m são pares, apresentamos o erro relativo (normalizado para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura numérica de algumas ordens das quadraturas discutidas nas seções 5.2.1−5.2.4, para $l, m = 0, 2, 4, \ldots, 50$. No apêndice C, apresentamos os

valores mínimo e máximo do erro relativo entre o valor analítico e o da quadratura numérica (na esfera unitária), nos casos em que l e m são pares, para algumas ordens de quadratura dos esquemas abordados nesta seção. Na seção seguinte analisamos o desempenho das quadraturas estudadas na integração numérica da equação (5.2).

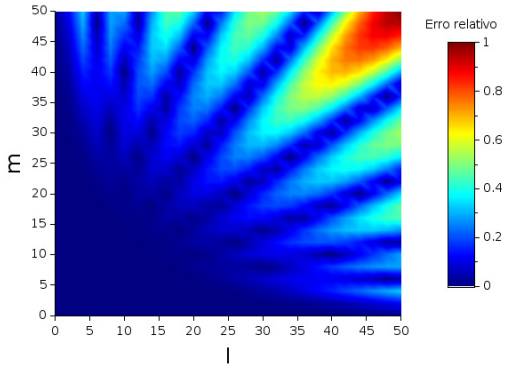


(a) $N = 2, M = 1$

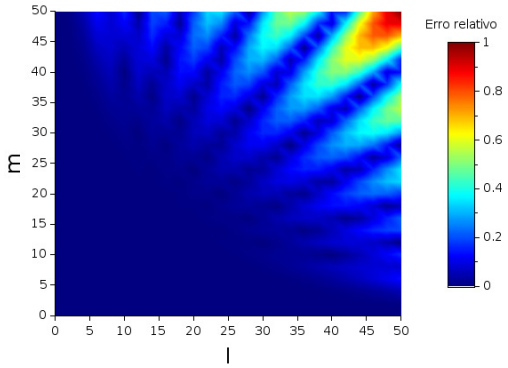
(b) $N = 4, M = 4$



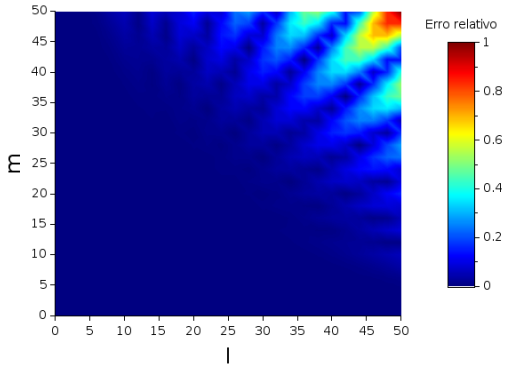
(c) $N = 8, M = 16$



(d) $N = 12, M = 36$

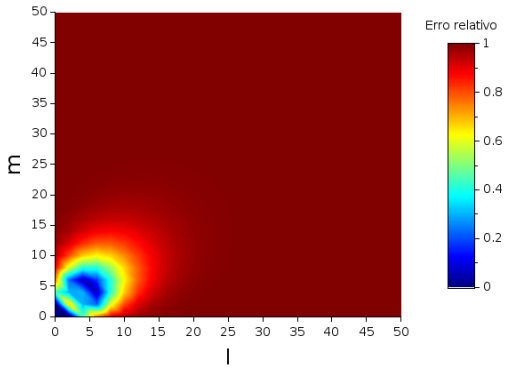


(e) $N = 16, M = 64$

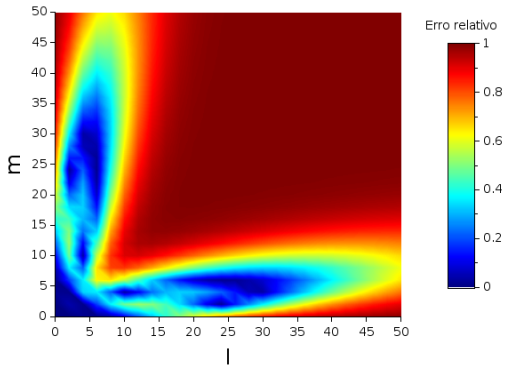


(f) $N = 20, M = 100$

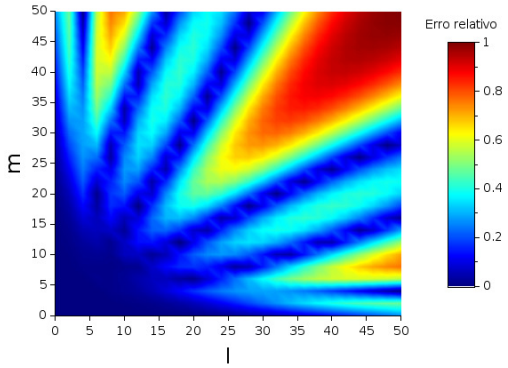
Figura 5.13 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $P_N T_N$, nos casos em que l e m são pares



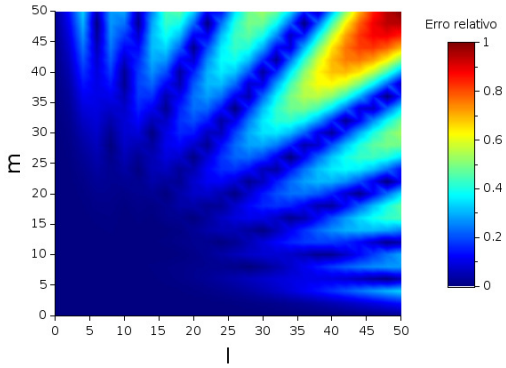
(a) $N = 2, M = 1$



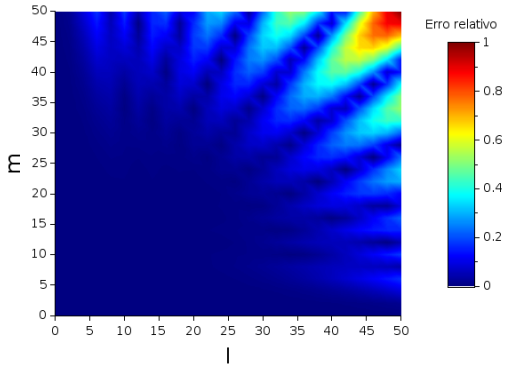
(b) $N = 4, M = 3$



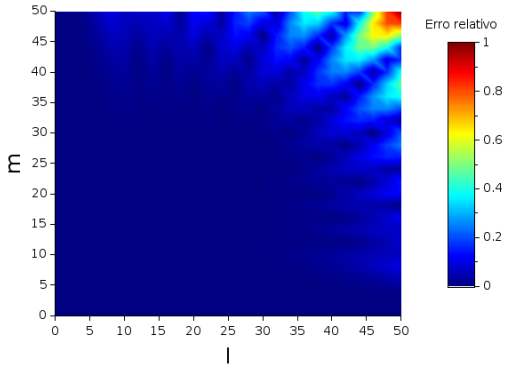
(c) $N = 8, M = 10$



(d) $N = 12, M = 21$

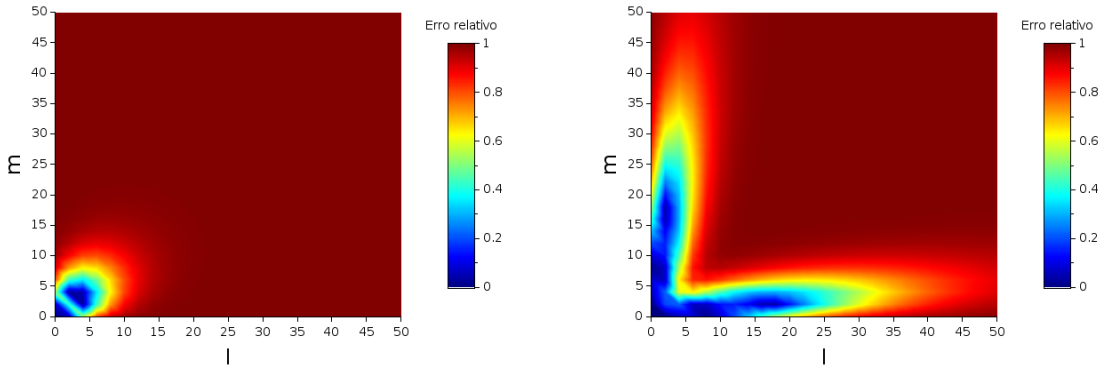


(e) $N = 16, M = 36$

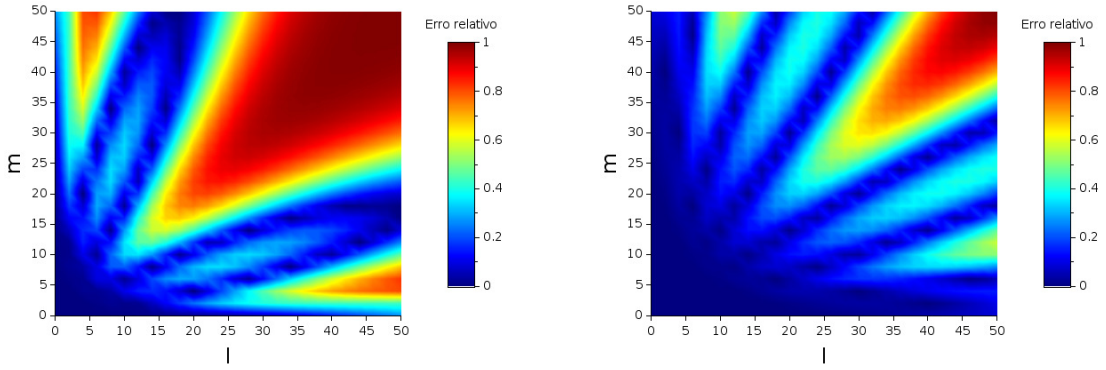


(f) $N = 20, M = 55$

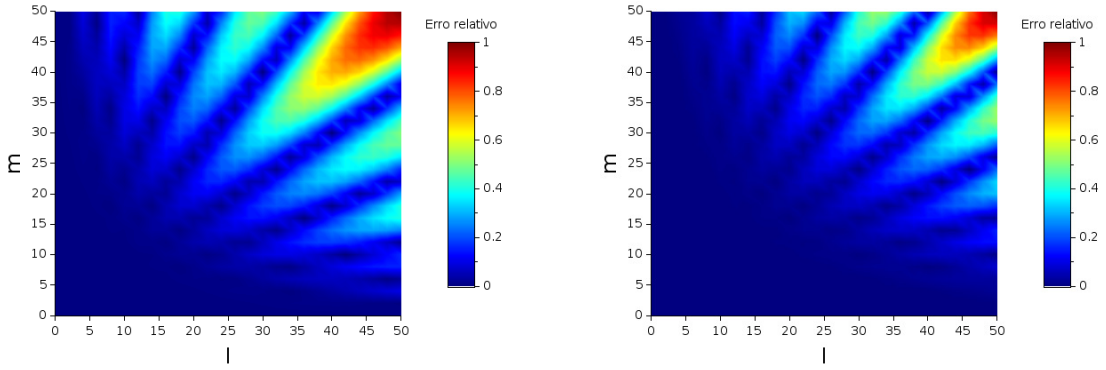
Figura 5.14 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $P_N T_N S_N$, nos casos em que l e m são pares



(a) $N_\theta = 1, M = 1$ (b) $N_\theta = 2, M = 4$

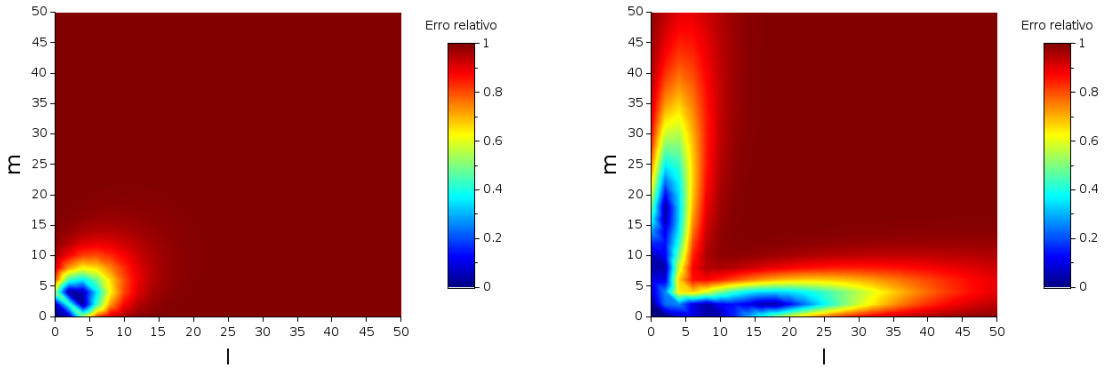


(c) $N_\theta = 4, M = 16$ (d) $N_\theta = 6, M = 36$



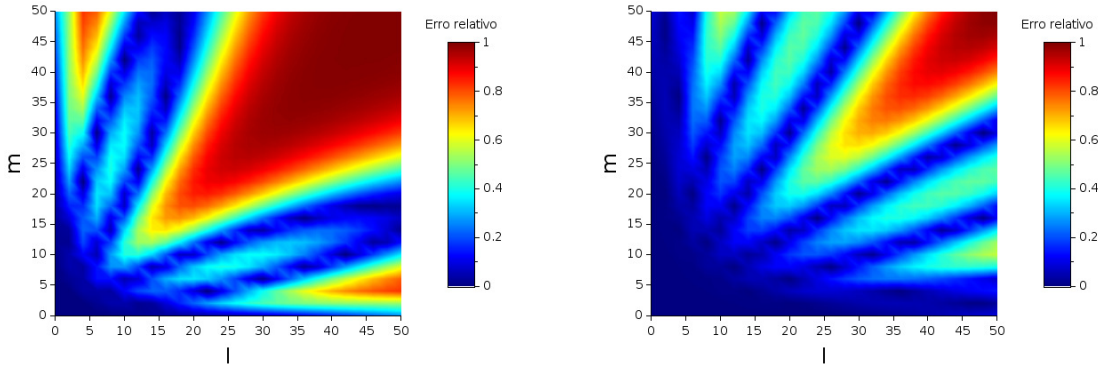
(e) $N_\theta = 8, M = 64$ (f) $N_\theta = 10, M = 100$

Figura 5.15 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRS45m\_Q_{N_\theta}$, nos casos em que l e m são pares



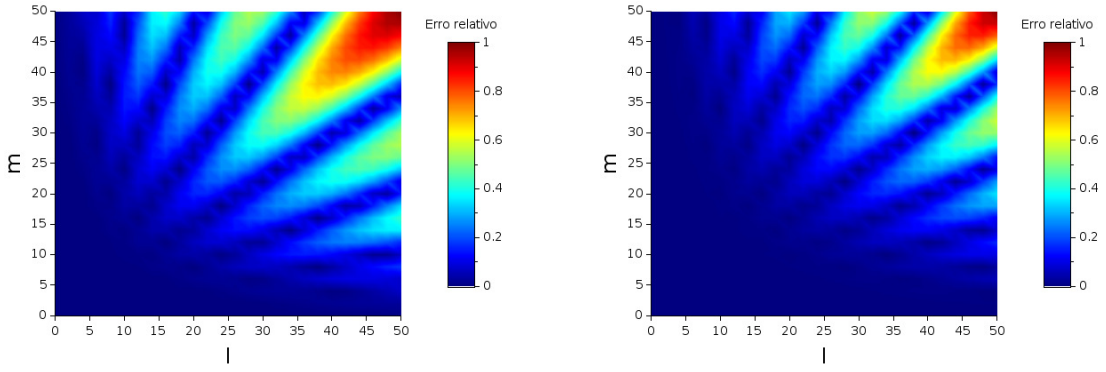
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 4$



(c) $N_\theta = 4, M = 16$

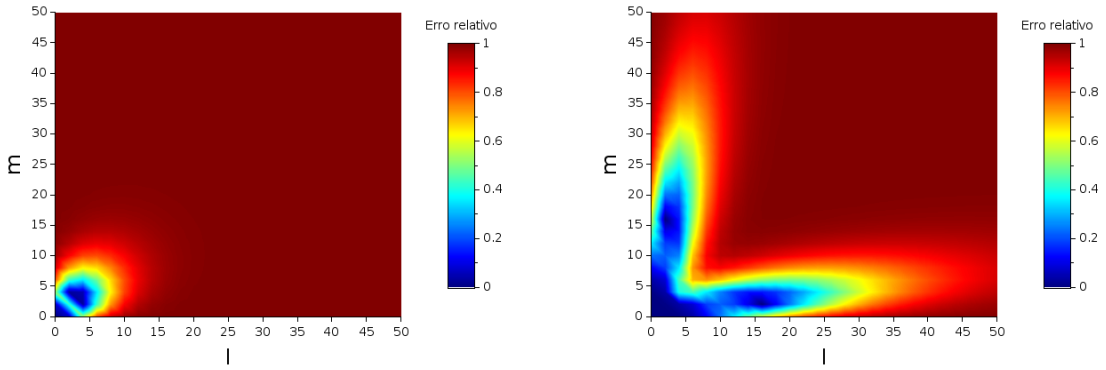
(d) $N_\theta = 6, M = 36$



(e) $N_\theta = 8, M = 64$

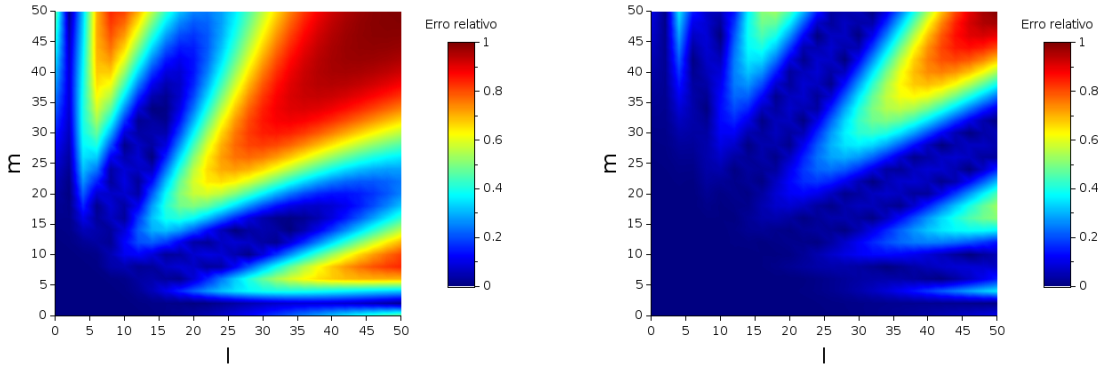
(f) $N_\theta = 10, M = 100$

Figura 5.16 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRA45m\_Q_{N_\theta}$, nos casos em que l e m são pares



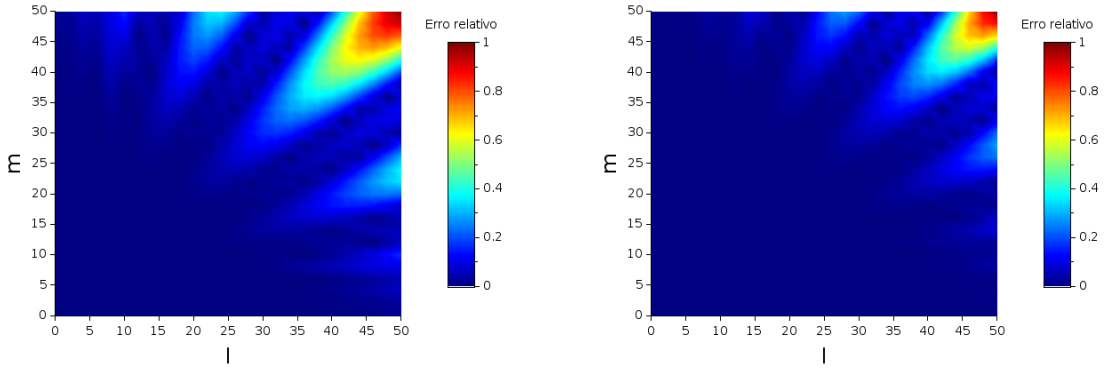
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 4$



(c) $N_\theta = 4, M = 16$

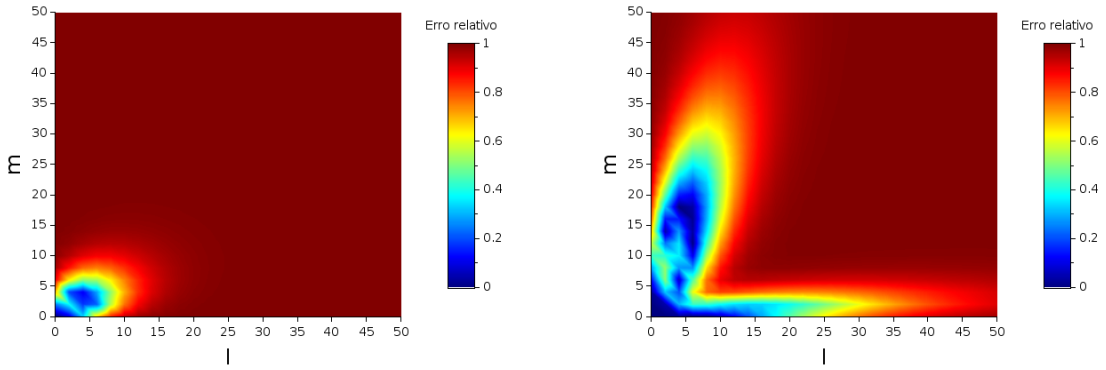
(d) $N_\theta = 6, M = 36$



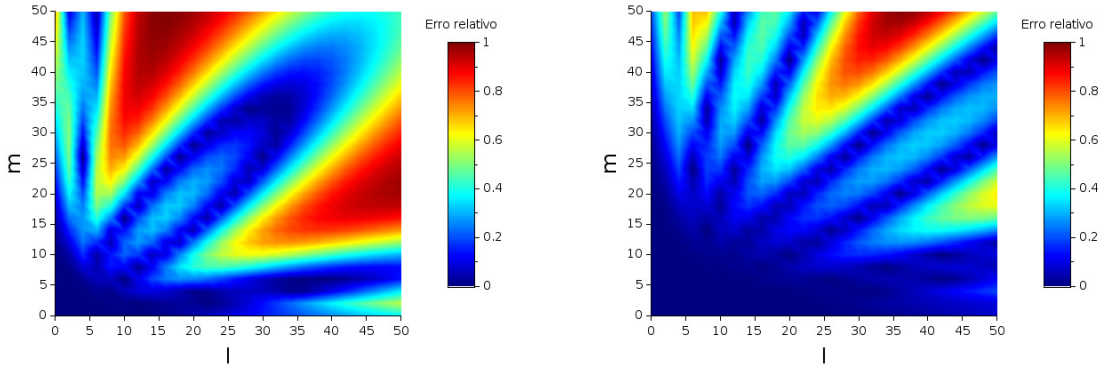
(e) $N_\theta = 8, M = 64$

(f) $N_\theta = 10, M = 100$

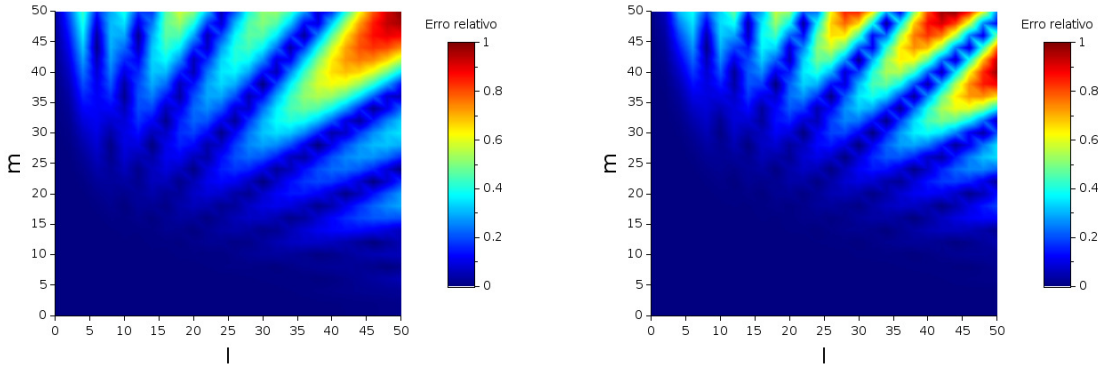
Figura 5.17 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRJ45m\_Q_{N_\theta}$, nos casos em que l e m são pares



(a) $N_\theta = 1, M = 1$ (b) $N_\theta = 2, M = 4$

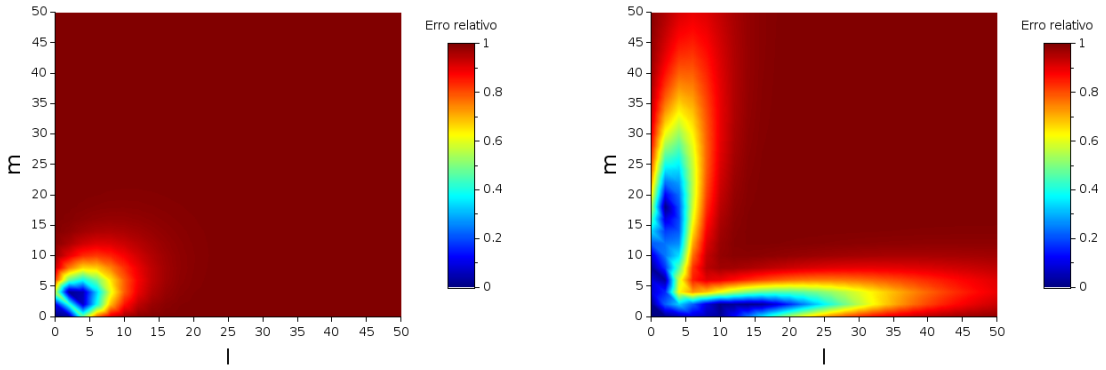


(c) $N_\theta = 4, M = 16$ (d) $N_\theta = 6, M = 36$



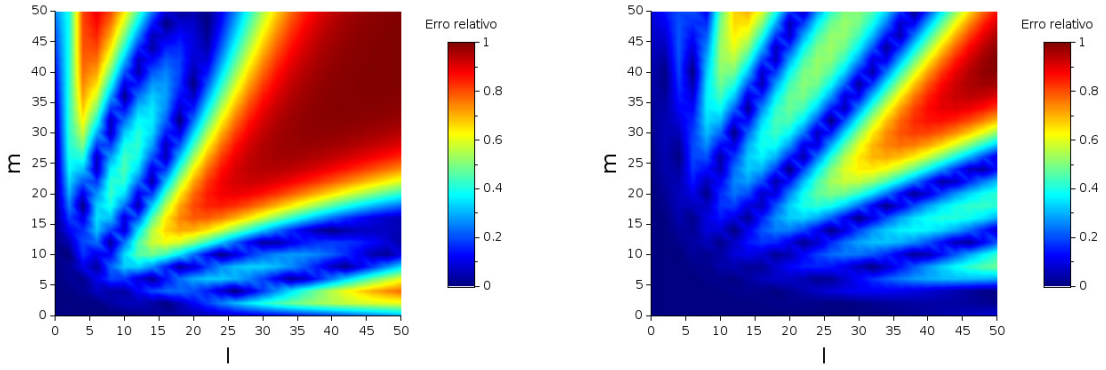
(e) $N_\theta = 8, M = 64$ (f) $N_\theta = 10, M = 100$

Figura 5.18 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRS90m\_Q_{N_\theta}$, nos casos em que l e m são pares



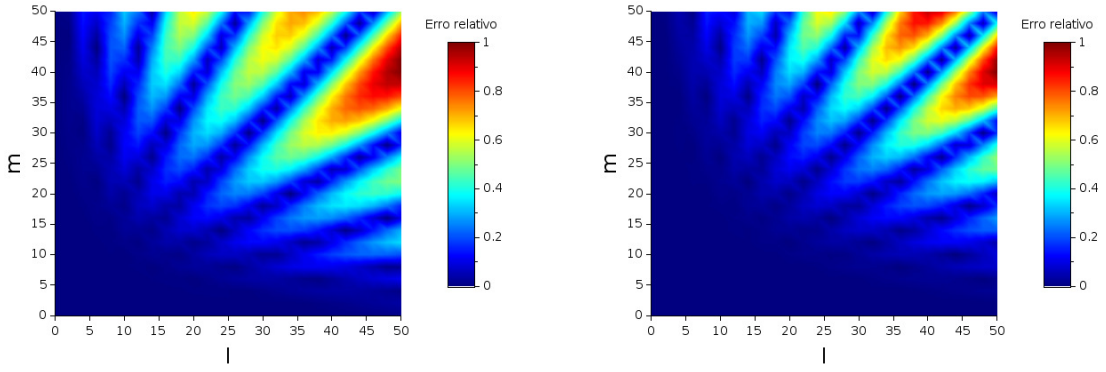
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 4$



(c) $N_\theta = 4, M = 16$

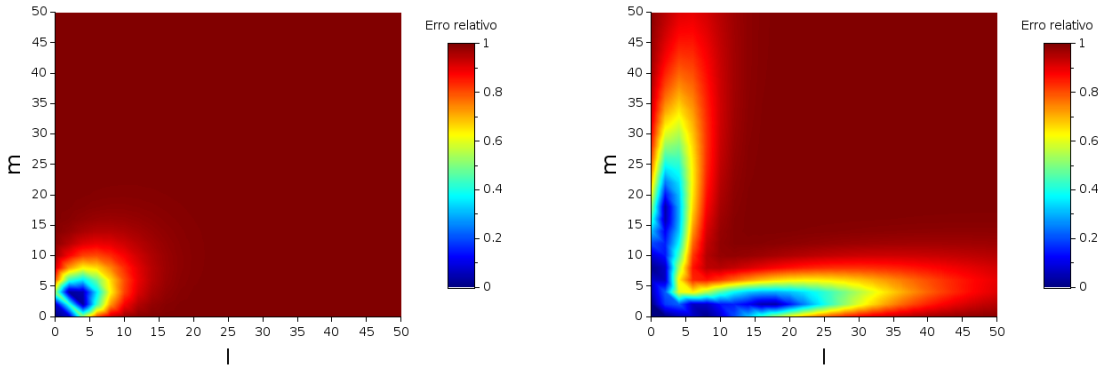
(d) $N_\theta = 6, M = 36$



(e) $N_\theta = 8, M = 64$

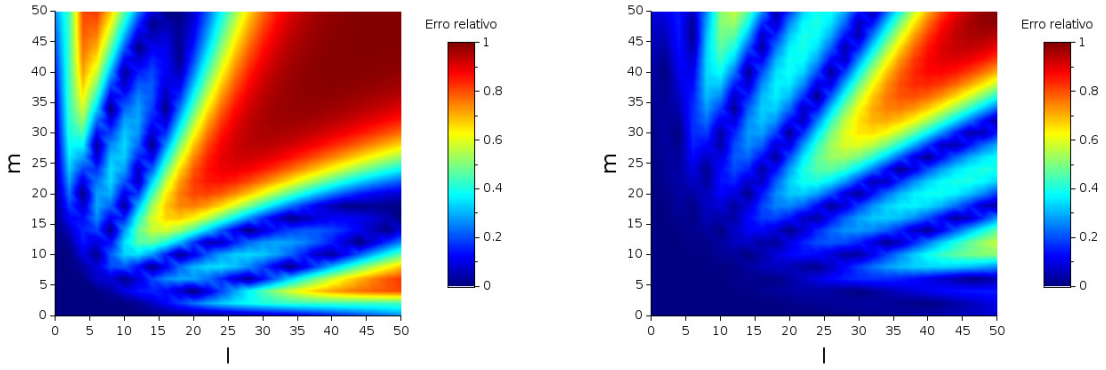
(f) $N_\theta = 10, M = 100$

Figura 5.19 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRJ90m\_Q_{N_\theta}$, nos casos em que l e m são pares



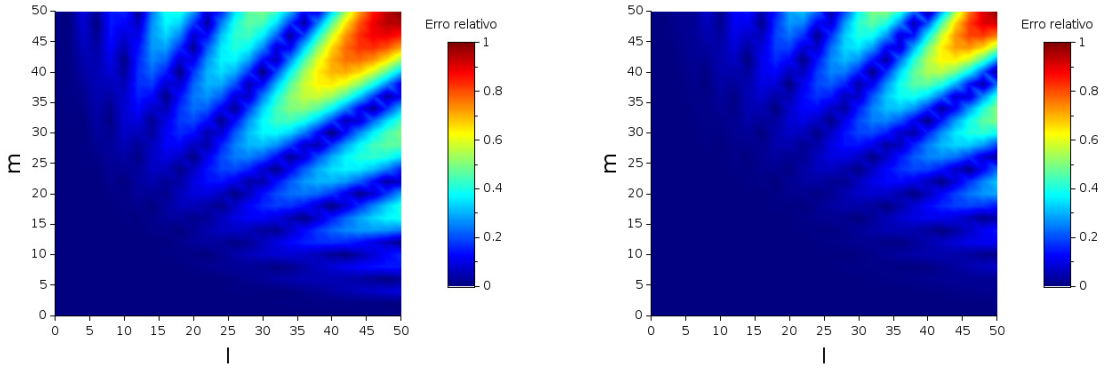
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 3$



(c) $N_\theta = 4, M = 10$

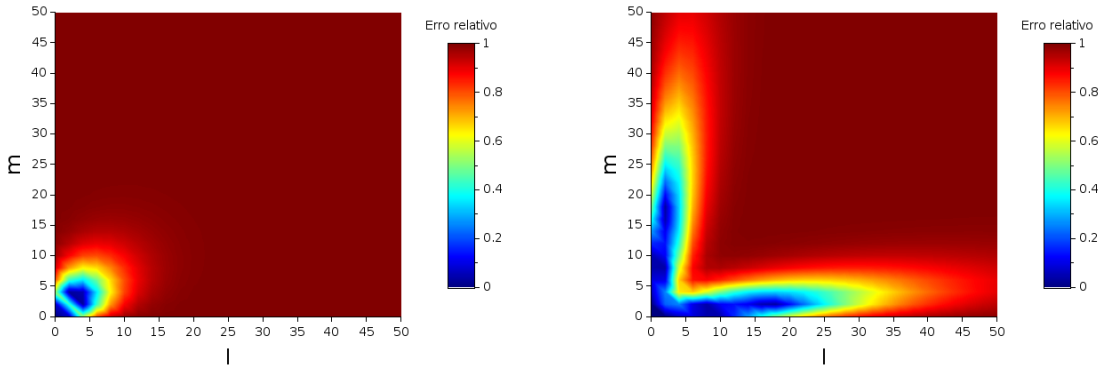
(d) $N_\theta = 6, M = 21$



(e) $N_\theta = 8, M = 36$

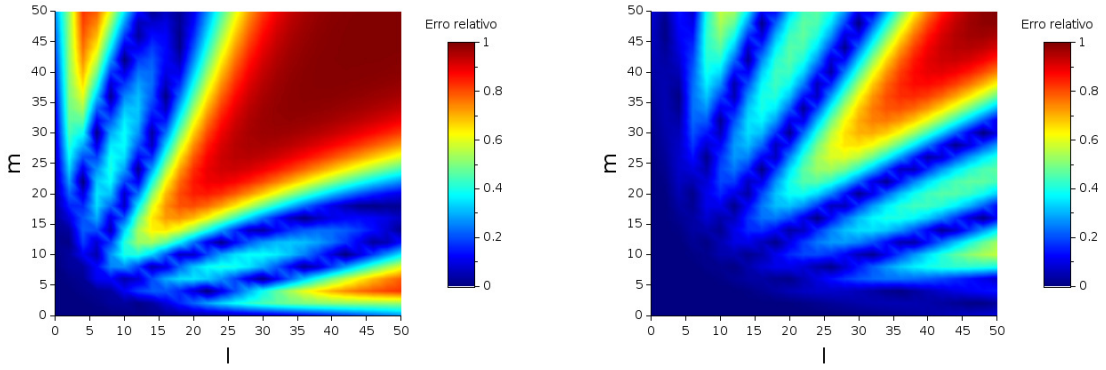
(f) $N_\theta = 10, M = 55$

Figura 5.20 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRS45m\_T_{N_\theta}$, nos casos em que l e m são pares



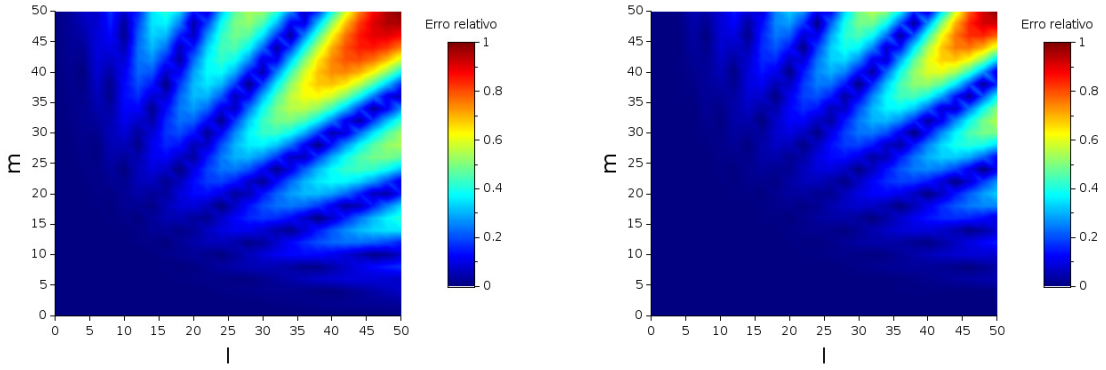
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 3$



(c) $N_\theta = 4, M = 10$

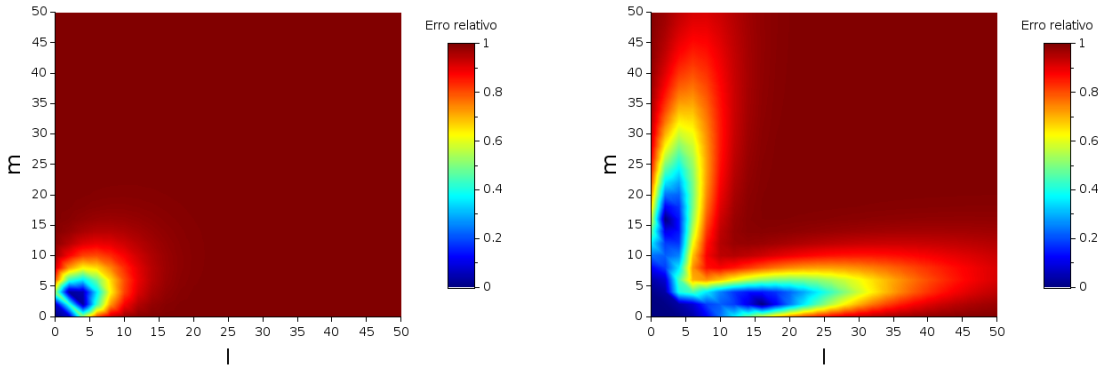
(d) $N_\theta = 6, M = 21$



(e) $N_\theta = 8, M = 36$

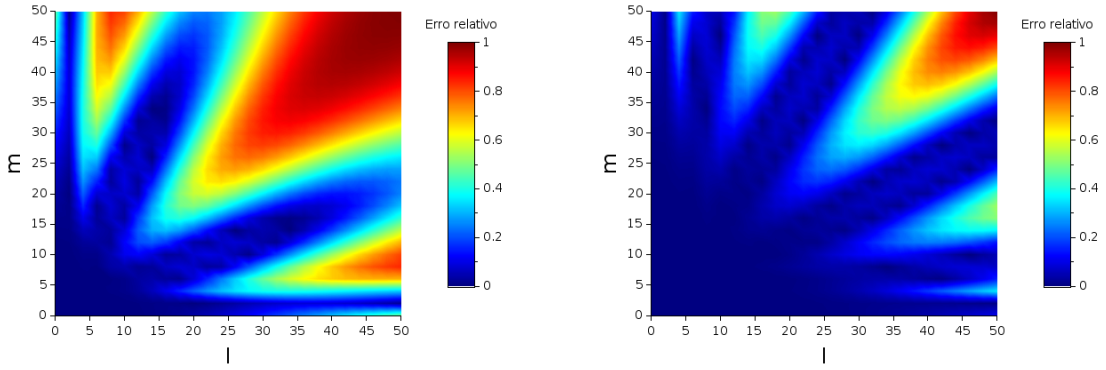
(f) $N_\theta = 10, M = 55$

Figura 5.21 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRA45m\_T_{N_\theta}$, nos casos em que l e m são pares



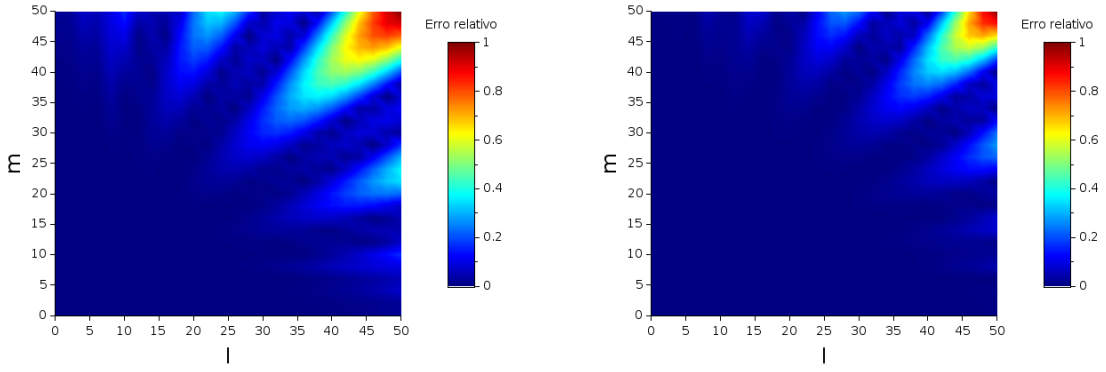
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 3$



(c) $N_\theta = 4, M = 10$

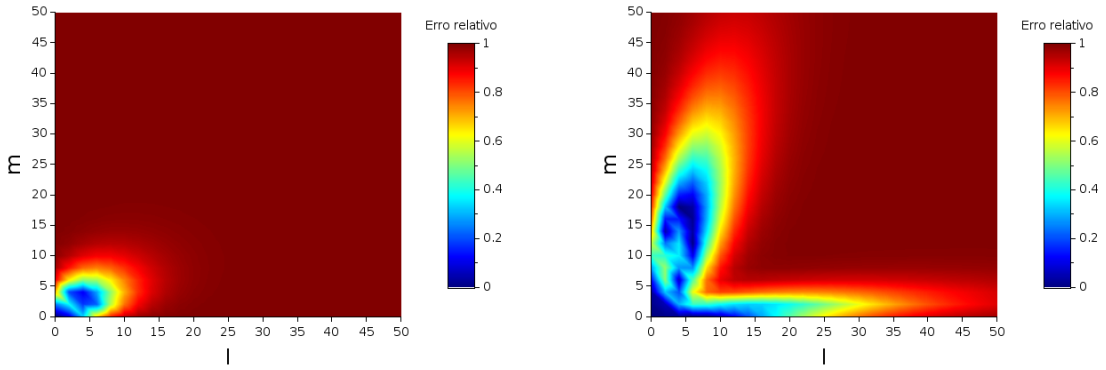
(d) $N_\theta = 6, M = 21$



(e) $N_\theta = 8, M = 36$

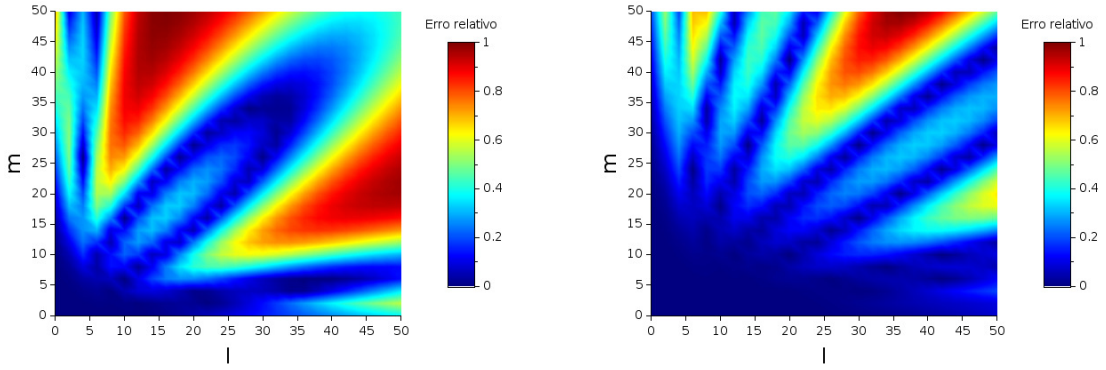
(f) $N_\theta = 10, M = 55$

Figura 5.22 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRJ45m\_T_{N_\theta}$, nos casos em que l e m são pares



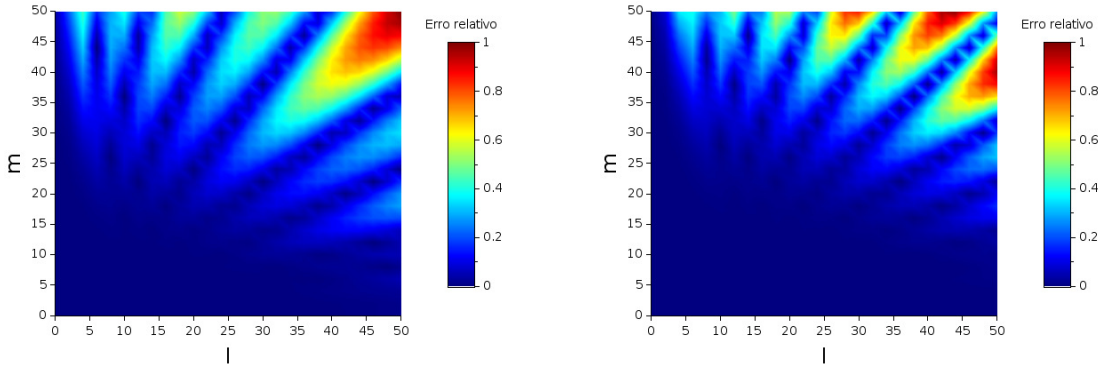
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 3$



(c) $N_\theta = 4, M = 10$

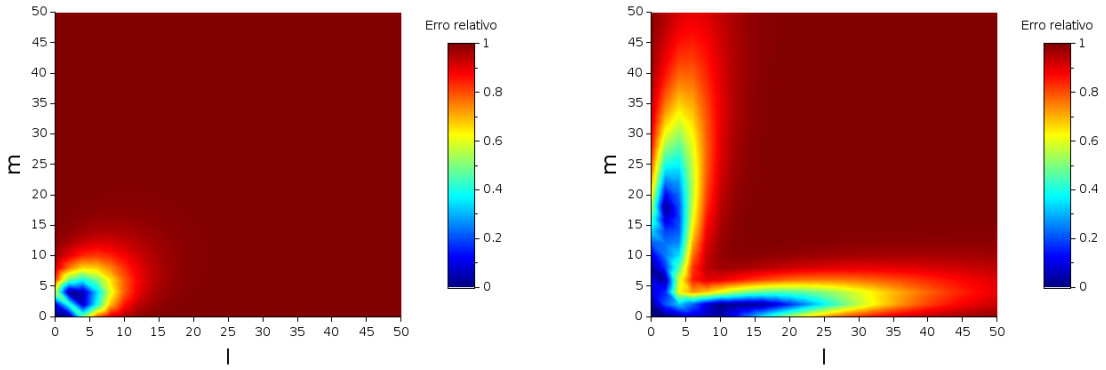
(d) $N_\theta = 6, M = 21$



(e) $N_\theta = 8, M = 36$

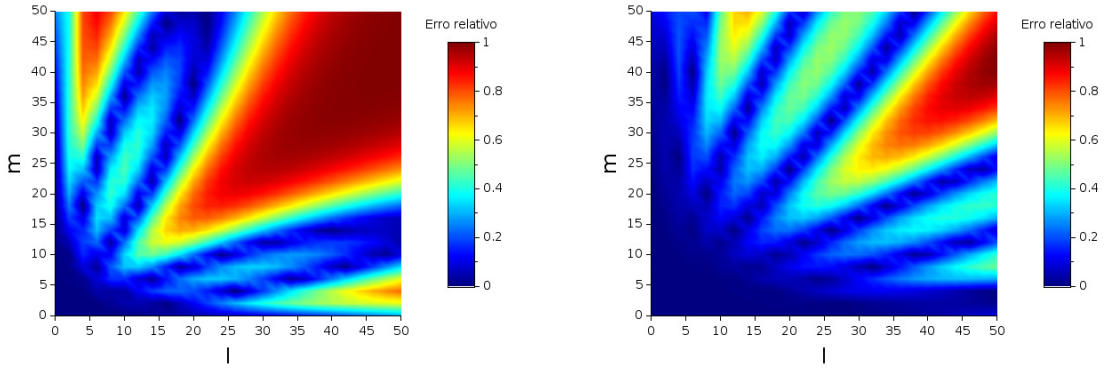
(f) $N_\theta = 10, M = 55$

Figura 5.23 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRS90m\_T_{N_\theta}$, nos casos em que l e m são pares



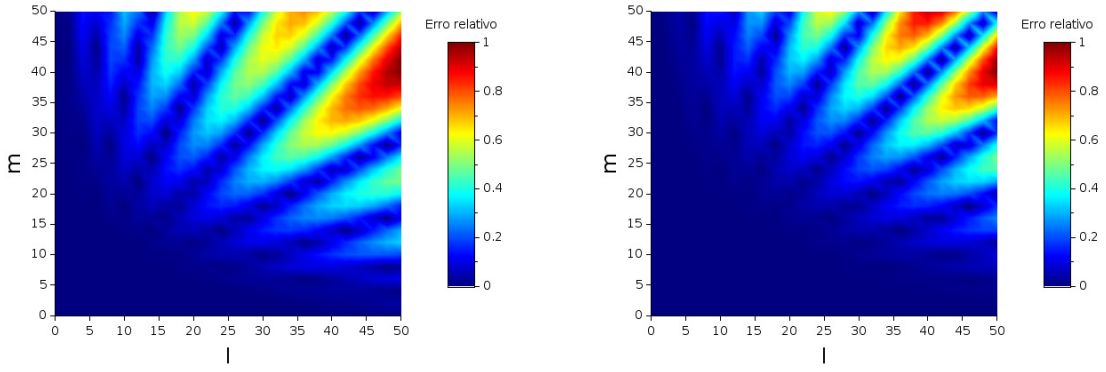
(a) $N_\theta = 1, M = 1$

(b) $N_\theta = 2, M = 3$



(c) $N_\theta = 4, M = 10$

(d) $N_\theta = 6, M = 21$



(e) $N_\theta = 8, M = 36$

(f) $N_\theta = 10, M = 55$

Figura 5.24 Erros relativos (normalizados para o intervalo $[0, 1]$) entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRJ90m\_T_{N_\theta}$, nos casos em que l e m são pares

5.3 Caso Particular

Nesta seção analisamos o desempenho dos esquemas de quadraturas $P_N T_N$, $P_N T_N S_N$ e QR na integração numérica da equação (5.2). Para tal, iniciamos analisando a regularidade da função $f(\theta, \phi)$ dada pela equação (5.3), e notamos que ela é uma função suave com respeito ao ângulo polar θ, de modo que ela possui $2N_\theta$ derivadas contínuas e limitadas (por Z_p) com respeito a θ, e concluímos que sua regularidade com respeito ao ângulo polar é

$$f(\theta, \phi) \subset \mathcal{C}_{Z_p}^{(2N_\theta)}. \tag{5.6}$$

De fato, a função (5.3) é contínua e limitada em todo o domínio angular, mas a primeira derivada em ϕ possui uma descontinuidade em $\phi = \pi/4$ e é limitada (por Z_a). Dessa forma, concluímos que a sua regularidade com respeito ao ângulo azimutal ϕ é

$$f(\theta, \phi) \subset \mathcal{C}_{Z_a}^{(1)}. \tag{5.7}$$

Uma vez que conhecemos as regularidades da função (5.3), podemos estimar a ordem de convergência do erro de truncamento da quadratura baseado nos resultados vistos nas seções 3.3 e 4.3, uma vez que todas as quadraturas estudadas neste trabalho são quadraturas produtos, e as respectivas quadraturas unidimensionais são quadraturas gaussianas.

Como dito na seção 3.3, nos casos em que a função $f(x) \subset \mathcal{C}_Z^r$ possui uma singularidade no interior do intervalo de integração, o erro de truncamento da quadratura gaussiana com n pontos converge como:

$$E_n(f) = \mathcal{O}(n^{-r-1}). \tag{5.8}$$

Sendo assim, temos que o erro de truncamento da quadratura azimutal converge como:

$$|E_{N_\phi}^{Q_a}[f]| = \mathcal{O}(N_\phi^{-2}), \tag{5.9}$$

em que $E_{N_\phi}^{Q_a}[f]$ representa o erro de truncamento da quadratura associada a variável azimutal com N_ϕ pontos. Por sua vez, como a quadratura polar não possui singularidade, o erro de truncamento da quadratura polar converge de acordo com a equação (3.70), isto é,

$$|E_{N_\theta}^{Q_p}[f]| = \mathcal{O}(N_\theta^{-2N_\theta}), \tag{5.10}$$

com $E_{N_\theta}^{Q_p}[f]$ representando o erro de truncamento da quadratura associada a variável polar com N_θ pontos.

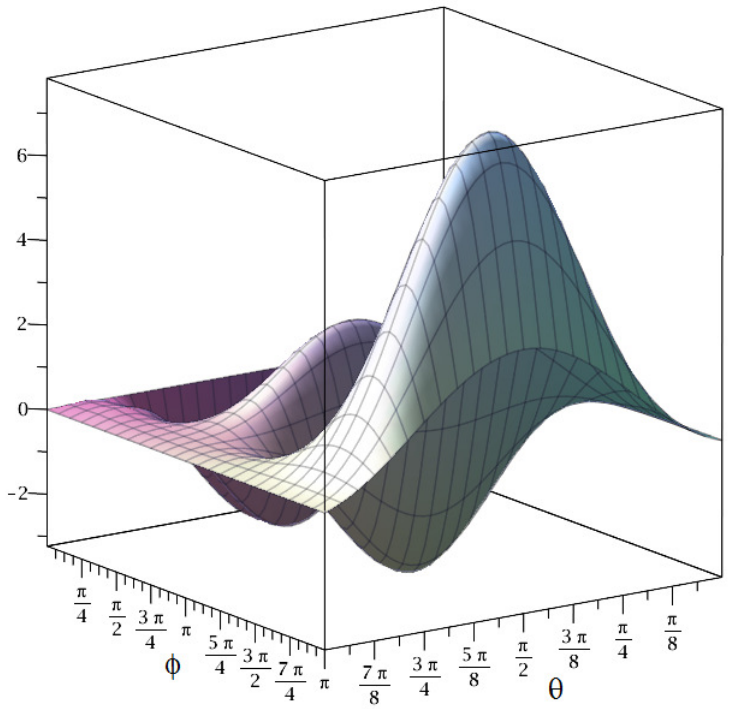


Figura 5.25 Gráfico da função (5.3)

Como $N_\phi = \mathcal{O}(N)$ e $N_\theta = \mathcal{O}(N)$, e na tabela 5.1 mostramos que o número de direções discretas por octante da esfera unitária M é de ordem $\mathcal{O}(N^2)$ para todas as quadraturas consideradas, da equação (4.93), concluímos que o erro de truncamento da quadratura produto converge como

$$|E_M^Q[f]| = \mathcal{O}(M^{-1}), \tag{5.11}$$

para todas as quadraturas estudadas. Nas figuras 5.26 e 5.27 apresentamos o gráfico do $|E|$,

$$E = \int_0^{\pi} \int_0^{2\pi} f(\theta, \phi) \sin\theta d\phi d\theta - \sum_{i=1}^{M_t} W_i f(\theta_i, \phi_i), \tag{5.12}$$

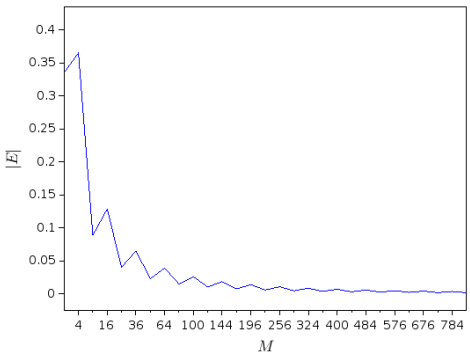
em função do número de direções, para os esquemas com disposição de direções quadrangular e triangular, respectivamente.

Observamos que o comportamento do erro seguiu a previsão feita, isto é, $E = \mathcal{O}(M^{-1})$, para todos os esquemas de quadratura. Também observamos um comportamento oscilatório no valor de E, que é explicado a seguir. Analisando o sinal de E, observamos que para as quadraturas $P_N T_N$, $QRS45m\_Q_{N_\theta}$, $QRA45m\_Q_{N_\theta}$ e $QRJ45m\_Q_{N_\theta}$, o sinal e E muda de acordo com o número de cones polares da quadratura, de modo que, as ordens de quadratura que possuem um número ímpar de cones polares apresentam E com sinal negativo, enquanto que as ordens de quadratura que possuem um número par de cones polares apresentam E com sinal positivo. Para estas quadraturas, também observamos que em módulo, as ordens com cones polares ímpares apresentam um valor de E maior do que as ordens com cones polares pares.

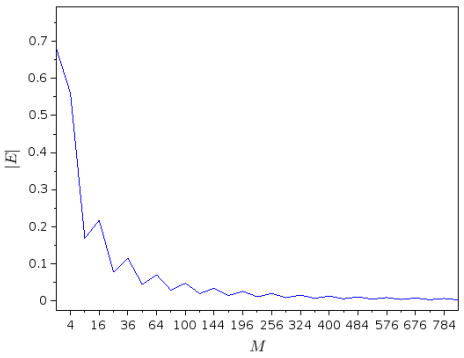
Para as quadraturas $P_N T_N S_N$, $QRS45m\_T_{N_\theta}$, $QRA45m\_T_{N_\theta}$ e $QRJ45m\_T_{N_\theta}$, não ocorre uma mudança de sinal no valor de E, mas observamos

que as ordens de quadratura com um número de cones polar par apresentam valores de E com menor magnitude (em módulo) do que as ordens com um número de cones polar ímpar.

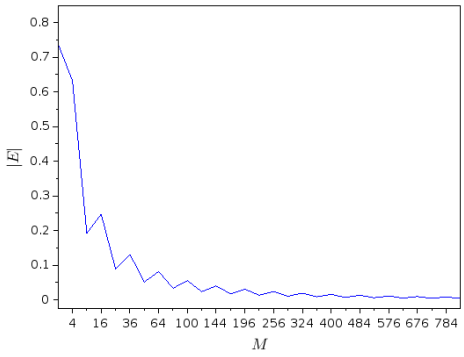
Nas demais quadraturas, observamos também esse caráter oscilatório em que o valor de E muda de sinal. Contudo, não identificamos um padrão para tal. Estes comportamentos oscilatórios também foram encontrados por Hu [43].



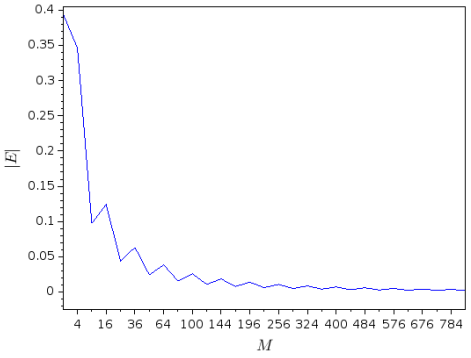
(a) $P_N T_N$



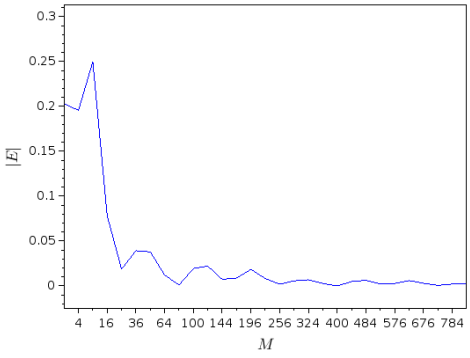
(b) $QRS45m\_Q_{N_\theta}$



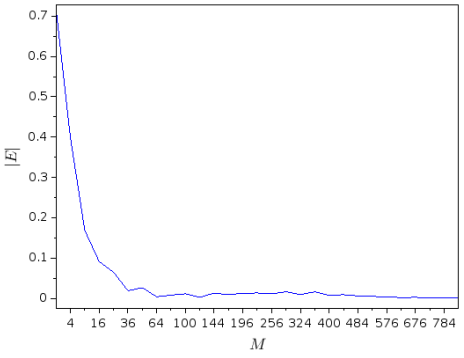
(c) $QRA45m\_Q_{N_\theta}$



(d) $QRJ45m\_Q_{N_\theta}$

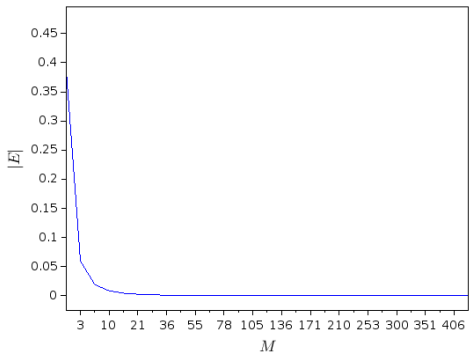


(e) $QRS90m\_Q_{N_\theta}$

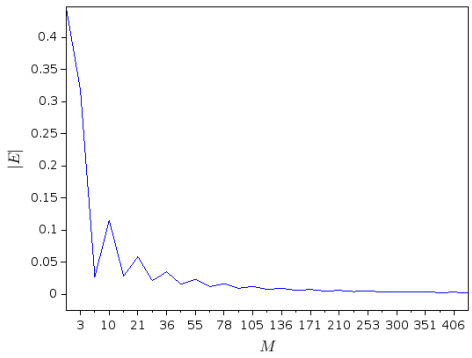


(f) $QRJ90m\_Q_{N_\theta}$

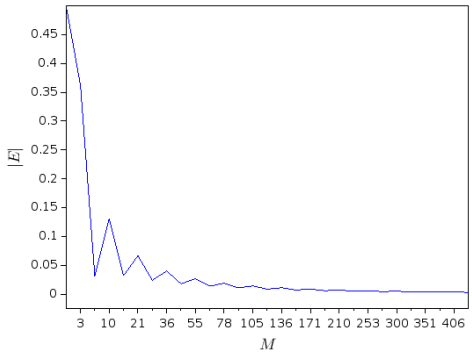
Figura 5.26 Erro absoluto para as quadraturas quadrangulares



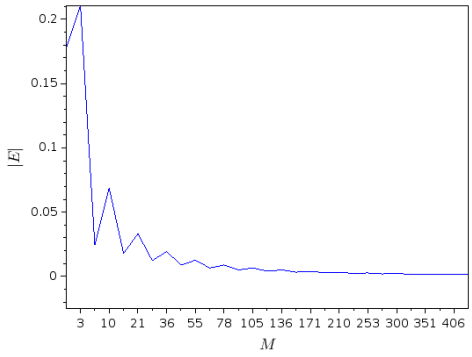
(a) $P_N T_N S_N$



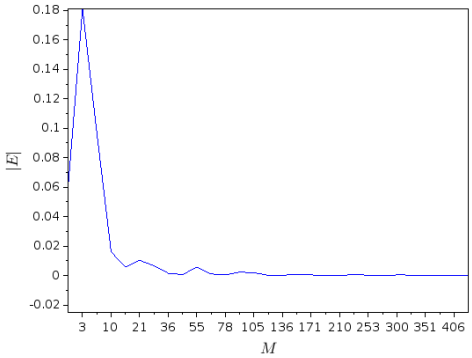
(b) $QRS45m\_T_{N_\theta}$



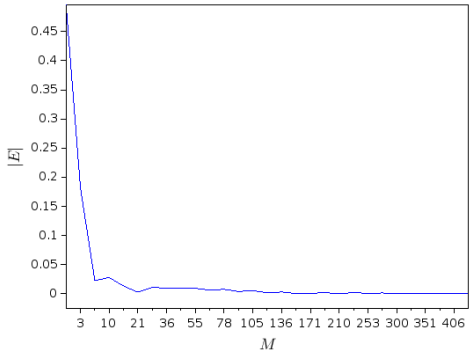
(c) $QRA45m\_T_{N_\theta}$



(d) $QRJ45m\_T_{N_\theta}$



(e) $QRS90m\_T_{N_\theta}$



(f) $QRJ90m\_T_{N_\theta}$

Figura 5.27 Erro absoluto para as quadraturas triangulares

6 CONCLUSÕES

Neste trabalho, estudamos aspectos teóricos e computacionais de quadraturas produto para integração na esfera unitária. Tais quadraturas resultam do produto de quadraturas gaussianas associadas a variável polar, θ, e azimutal, ϕ. Polinômios ortogonais assumem um papel importante na obtenção dos nós e pesos destas quadraturas, pois através deles transformamos o cálculo dos nós de quadratura em um problema de autovalores de uma matriz tridiagonal simétrica, evitando a resolução de sistemas não lineares (abordagem clássica) que se mostram mal-condicionados e não capazes de gerar quadraturas de alta ordem.

As quadraturas P_NT_N e $P_NT_NS_N$ são construídas através das quadraturas unidimensionais de Gauss-Legendre e Gauss-Chebyshev, e se diferenciam pela forma como são combinadas as ordens das quadraturas unidimensionais utilizadas, bem como a escolha dos pesos da quadratura produto. Com isso, a quadratura P_NT_N assume um padrão quadrangular, enquanto a quadratura $P_NT_NS_N$ assume um padrão triangular de disposição das direções discretas na esfera unitária.

Para as as quadraturas QR, estudamos os aspectos teóricos e computacionais tanto da abordagem proposta por Abu-Shumays [1, 2] via sistemas não lineares, quanto da abordagem proposta por Spence [70] via polinômios ortogonais. Destacamos as dificuldades na obtenção das quadraturas QR via sistemas não lineares, de modo que conseguimos reproduzir poucos resultados através desta abordagem, seguindo o que já visto na literatura. Além das quadraturas QR terem o problema na sua geração (mal condicionamento dos sistemas não lineares), estudos anteriores na resolução da equação de transporte [73] tem apresentado melhores resultados

em relação ao efeito raio com a utilização deste esquema de quadratura. Devido a isso, analisamos de forma mais particular as quadraturas QR, e em particular, na abordagem proposta por Spence (via polinômios ortogonais). Baseados nesta proposta, propusemos dois conjuntos de quadraturas para a variável azimutal, sendo que um deles tem implementação computacional muito simples, pois não é necessário a resolução de problemas de autovalores, e quando consideramos a integração na esfera unitária, demonstrou um desempenho computacional superior às demais quadraturas QR. Destacamos que, neste trabalho, seguindo a forma como é construída a quadratura QR, somente um esquema de quadratura unidimensional é associado a variável polar, e cinco esquemas de quadratura unidimensional são associados a variável azimutal.

Como a existência de singularidades tem extrema relevância no comportamento assintótico do erro de truncamento, propusemos um conjunto de quadratura produto para o caso em que temos uma singularidade azimutal em $\phi = \phi_0$. Em nossa proposta, a quadratura para a variável azimutal é gerada de modo que o ponto de singularidade nunca é nó da quadratura, e a obtenção dos nós e pesos de quadratura é feita seguindo a abordagem via polinômios ortogonais.

Além disso, estimativas para o erro de truncamento dos esquemas de quadratura, tanto das quadraturas gaussianas unidimensionais quanto das quadraturas produto foram apresentadas. Em particular, apresentamos estimativas para a ordem de convergência das quadraturas produto em função do número de direções discretas por octante da esfera unitária.

Implementamos em Fortran 95 códigos para a geração destes esquemas de quadratura multidimensionais com ordem arbitrária em precisão simples, dupla

e quádrupla. Tais códigos são baseados na teoria dos polinômios ortogonais, e destacamos o uso tanto da quadratura *função erro* quanto da quadratura *tanh-sinh* na obtenção das entradas da matriz tridiagonal necessária para a obtenção dos nós de quadratura. Baseados no algoritmo proposto por Spence [70] e utilizando as quadraturas *função erro* e *tanh-sinh*, foi possível a implementação de algoritmos para a geração de quadraturas QR de altas ordens, o que é um ganho, visto que somente esquemas de baixa ordem foram gerados numericamente [2]. Os códigos implementados demonstraram um baixo tempo computacional na geração destas quadraturas. Um dos motivos para isto é que não calculamos autovetores para a obtenção dos pesos de quadratura, uma vez que eles são obtidos através de um somatório. A implementação em precisão quádrupla nos garante que conseguimos gerar os conjuntos de quadratura com a maior precisão possível usando a tecnologia de hardware disponível em computadores de uso geral, sem ter de recorrer a esquemas de simulação de precisão múltipla em software.

Em relação a forma como acoplamos as quadraturas unidimensionais, o acoplamento quadrangular, tanto para as quadraturas QR quanto a P_NT_N, demonstraram um desempenho superior ao acoplamento triangular. Ambos acoplamentos apresentaram resultados com precisão semelhante, mas através do acoplamento quadrangular é possível gerar quadraturas produto com um grande número de direções discretas por octante da esfera unitária utilizando baixas ordens de quadratura.

Também analisamos o desempenho destes esquemas tanto no octante principal quanto na esfera unitária. Para o octante principal, percebemos que os esquemas de quadratura que utilizam a quadratura de Gauss-Chebyshev para a variável azimutal (P_NT_N, $P_NT_NS_N$ e $QRJ45m$) não são capazes de integrar com precisão os casos em que l e m são ímpares e menores que 10, e destacamos o desempenho da

quadratura $QRS45m\_Q$ e $QRS45m\_T$, que foi capaz de integrar os valores testados com erros relativos de ordem $\mathcal{O}(10^{-8})$ ou menores. Já para a esfera unitária, concluímos que todos os esquemas são adequados para integrar a equação (2.48) para os valores de l ou m ímpar. Para os casos em que l e m são pares, a quadratura P_NT_N perfez o melhor desempenho numérico, seguida da quadratura $QRJ45m\_Q$. Concluímos também que as quadraturas $QRS45m\_Q$ e $QRS45m\_T$ possuem desempenho superior às quadraturas $QRA45m\_Q$ e $QRA45m\_T$, respectivamente, e também, apesar do desempenho não satisfatório das quadraturas $QRJ45m\_Q$ e $QRJ45m\_T$ para o octante principal, elas são boas alternativas para a integração na esfera unitária devido aos resultados apresentados e simples geração da quadratura azimutal. Tanto no octante principal quanto na esfera unitária, os esquemas QR baseados na quadratura *Azimutal-QRS*45 mostraram resutados mais precisos e tempos computacionais semelhentes aos esquemas baseados na quadratura *Azimutal-QRA*45.

Baseados em todo o estudo, devido a ao baixo tempo computacional e precisão nos resultados obtidos, os esquemas mais indicados para a integração no octante principal são os esquemas $QRS45m\_Q_{N_\theta}$ seguido do esquema $QRA45m\_Q_{N_\theta}$. Já para a esfera unitária, os esquemas mais indicados são os P_NT_N e $QRJ45m\_Q_{N_\theta}$. Destacamos que, integrar com precisão polinômios de graus elevados não é suficiente para determinar uma solução precisa para o problema de transporte.

Com base nos resultados aqui apresentados, alguns tópicos surgem como interesse na continuidade desse estudo, entre eles, a sequência natural de utilização destes esquemas precisos na solução de problemas multidimensionais de transporte de nêutrons, bem como análise assintótica de erro associadas ao método de ordenadas discretas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Abu-Shumays, I. K. Compatible product angular quadrature for neutron transport in x-y geometry. *Nuclear Science and Engineering v. 64* (1977), p.299–316.

[2] Abu-Shumays, I. K. Angular quadratures for improvid transport computations. *Transport Theory and Statistical Physics v. 30* (2001), p.169–204.

[3] Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. *LAPACK Users' Guide: Third Edition*. Philadelphia: SIAM, 1999.

[4] Atkinson, K. E. *An Introduction to Numerical Analysis*. New York: John Wiley & Sons, 1989.

[5] Badruzzaman, A. An efficient algorithm for nodal-transport solutions in multidimensional geometry. *Nuclear Science and Engineering v. 89*, n. 3 (1985), p.281–290.

[6] Bailey, D. H., Jeyabalan, K., and Li, X. S. A comparison of three high-precision quadrature schemes. *Experimental Mathematics v. 14*, n. 3 (2005), p.317–329.

[7] Barichello, L. B. Explicit formulations for radiative transfer problems. In *Thermal Measurements and Inverse Techniques*, H. R. B. Orlande, O. Fudyin, D. Maillet, and R. M. Cotta, Eds. CRC Press, Boca Raton, 2011. p.541-562.

[8] Barichello, L. B., and Siewert, C. A discrete-ordinates solution for a non-grey model with complete frequency redistribution. *Journal of Quantitative Spectroscopy and Radiative Transfer v. 62*, n. 6 (1999), p.665–675.

[9] Barichello, L. B., Tres, A., Picoloto, C. B., and Azmy, Y. Y. Recent Studies on the Asymptotic Convergence of the Spatial Discretization for Two-Dimensional Discrete Ordinates Solutions. *Journal of Computational and Theoretical Transport v. 45*, n. 4 (2016), p.299–313.

[10] Barros, R. C. D., and Larsen, E. W. A spectral nodal method for one-group X, Y-geometry discrete ordinates problems. *Nuclear Science And Engineering v. 111*, n. 1 (1992), p.34–45.

[11] Bell, G. I., and Glasstone, S. *Nuclear Reactor Theory*. New York: Van Nostrand Reinhold, 1970.

[12] Boltzmann, L. Weitere studien über das wärmegleichgewicht unter gas molekülen. *Sitzungsberichte Akademie der Wissenschaften v. 60* (1872), p.275–376.

[13] Borwein, J. M., Bailey, D. H., and Girgensohn, R. *Experimentation in Mathematics: Computational Paths to Discovery*. Taylor & Francis, 2004.

[14] Brass, H., and Petras, K. *Quadrature Theory: The Theory of Numericl Integration on a Compact Interval*. Providence: American Mathematical Society, 2011.

[15] Cacuci, D. G. *Handbook of Nuclear Engineering*. Boston: Springer, 2010.

[16] Case, K. M. Elementary solutions of the transport equation and their applications. *Annals of Physics v. 9*, n. 1 (1960), p.1–23.

[17] Chandrasekhar, S. *Radiative Transfer*. London: Oxford University Press, 1950.

[18] Da Cunha, R. D. *Programação científica em Fortran 95*, 2 ed. Porto Alegre: Edição do Autor, 2011.

[19] Da Cunha, R. D. *Comunicação privada*, 2019.

[20] Da Cunha, R. D., Tres, A., and Barichello, L. B. A study on the parallel, iterative solution of systems of linear equations appearing on analytical nodal schemes for two-dimensional cartesian geometry discrete ordinates problems. In *ANS MC2015 - Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method, 2015, Nashville, TN* (2015), ANS MC2015 - Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method. La Grange Park, IL: American Nuclear Society, 2015, p. 1-12.

[21] Datta, B. N. *Numerical Linear Algebra and Applications*. Pacific Grove: Brooks/Cole, 1995.

[22] Davis, P. J. *Interpolation and Approximation*. New York: Dover Publications Inc, 1975.

[23] Davis, P. J., and Rabinowitz, P. *Methods of Numerical Integration*. New York: Academic Press, 1984.

[24] de Almeida Konzen, P. H., Guidi, L. F., and Richter, T. Quasi-random discrete ordinates method for neutron transport problems. *Annals of Nuclear Energy v. 133*, n. 8 (2019), p.275–282.

[25] Debnath, L., and Bhatta, D. *Integral Transforms and their Applications*. Boca Raton: CRC Press, 2015.

[26] Duderstadt, J. J., and Hamilton, L. J. *Nuclear Reactor Analysis*. New York: John Wiley & Sons, 1976.

[27] ESI Group. Scilab. Version 6.0.2, 2019. Disponível em: https://www.scilab.org/download/6.0.2. Acesso em: 6 dez. 2019.

[28] Fletcher, J. K. The solution of the time-independent multi-group neutron transport equation using spherical harmonics. *Annals of Nuclear Energy v. 4* (1977), p.401–405.

[29] Fletcher, J. K. A solution of the neutron transport equation using spherical harmonics. *Journal Of Physics A: Mathematical and General v. 16*, n. 12 (1983), p.2827–2835.

[30] Franco, N. B. *Cálculo Numérico*. São Paulo: Pretence Hall, 2006.

[31] Garcia, R. D. M. Métodos para solução da equação de transporte de partículas integro-diferencial. Escola de verão em teoria de transporte de partículas nêutras, PUCRS, Porto Alegre., 2002.

[32] Garcia, R. D. M., and Siewert, C. E. Channel flow of a binary mixture of rigid spheres described by the linearized Boltzmann equation and driven by temperature, pressure and density gradients. *SIAM Journal on Applied Mathematics v. 67* (2007), p.1041–1063.

[33] Garcia, R. D. M., Siewert, C. E., and Yacout, A. Radiative transfer in a multilayer medium subject to fresnel boundary and interface conditions and uniform illumination by obliquely parallel roys. *Journal of Quantitative Spectroscopy and Radiative Transfer v. 109* (2008), p.2151–2170.

[34] Gautschi, W. *Numerical Analysis*, 2 ed. [S.l.]: Birkhäuser Basel, 2012.

[35] GCC. Gfortran. Version 7.4.0, 2018. Disponível em: https://gcc.gnu.org/wiki/GFortran. Acesso em: 6 dez. 2019.

[36] Golub, G., and Van Loan, C. *Matrix Computations*, 4 ed. Baltimore: The Johns Hopkins University Press, 2013.

[37] Golub, G. H., and Meurant, G. *Matrices, Moments and Quadrature with Applications*. Princeton: Princeton University Press, 2010.

[38] Golub, G. H., and Welsch, J. H. Calculation of Gauss quadrature rules. *Mathematics of Computation v. 23*, n. 106 (Apr. 1969), p.221–230+s1–s10.

[39] Gonzáles, M., Fernández, P., and Velarde, C. G. 2d numerical comparison between S_N and M_1 radiation transport methods. *Annals of Nuclear Energy v. 36* (2009), p.886–895.

[40] Haber, S. Numerical evaluation of multiple integrals. *SIAM Review v. 12*, n. 4 (Oct. 1970), p.481–526.

[41] Haghighat, A. *Monte Carlo Methods for Particle Transport*. [S.l.]: CRC Press, 2014.

[42] Hébert, A. High-order diamond differencing schemes. *Annals Of Nuclear Energy v. 33* (2006), p.1479–1488.

[43] Hu, X. *Asymptotic Convergence of the Angular Discretization Error in the Discrete Ordinates Approximation of the Particle Transport Equation*. PhD thesis, North Carolina State University, 2018.

[44] Hussein, E. M. A. *Computed Radiation Imaging: Physics and Mathematics of Forward and Inverse Problems*. Elsevier, 2011.

[45] JTC1/SC22/WG5. Fortran 95. ISO/IEC 1539:1997, 1997.

[46] Kelley, C. T. *Iterative Methods for Linear and Nonlinear Equations*. Philadelphia: SIAM, 1995.

[47] Klose, A. D., and Hielscher, A. H. Optical tomography with the equation of radiative transfer. *International Journal of Numerical Methods for Heat & Fluid Flow v. 18*, n. 3/4 (2008), p.443–464.

[48] Knackfuss, R. F., and Barichello, L. B. On the temperature-jump problem in rarefied gas dynamics: The effect of the cercignani-lampis boundary condition. *SIAM Journal on Applied Mathematics v. 66* (2006), p.2149–2186.

[49] Kópházi, J., Lathouwers, D., and Kloosterman, J. L. Development of a three-dimensional time-dependent calculation scheme for molten salt reactors and validation of the measurement data of the molten salt reactor experiment. *Nuclear Science and Engineering v. 163* (2009), p.118–131.

[50] Kress, R. *Numerical Analysis*. New York: Springer, 1991.

[51] Kütz, M. Asymptotic error bounds for a class of interpolatory quadratures. *Siam Journal On Numerical Analysis v. 21*, n. 1 (Feb. 1984), p.167–175.

[52] Lathrop, K. D. Remedies for ray effects. *Nuclear Science and Engineering v. 45*, n. 3 (1971), p.255–268.

[53] Lathrop, K. D., and Brinkley, F. W. *TWOTRAN-II, Two- dimensional multigroup discrete ordinates transport code*. Los Alamos: Los Alamos Scientific Laboratory Report, 1973.

[54] Lathrop, K. D., and Carlson, B. G. *Discrete Ordinate Angular Quadrature of the Neutron Transporte Equation*. Los Alamos Scientific Laboratory Report, LA-3186, Los Alamos, NM, 1965.

[55] Lawrence, R. Progress in nodal methods for the solution of the neutron diffusion and transport equations. *Progress in Nuclear Energy v. 17*, n. 3 (1986), p.271–301.

[56] LeVeque, R. J. *Finite Volume Methods for Hyperbolic Problems*. [S.l.] Cambridge University Press, 2002.

[57] Lewis, E., and Miller, W. *Computational Methods of Neutron Transport*. New York: John Wiley & Sons, 1984.

[58] Lewis, E. E., Miller, W. F., and Henry, T. P. A two-dimensional finite element method for integral neutron transport calculations. *Nuclear Science and Engineering v. 58*, n. 2 (1975), p.203–212.

[59] Lima, E. L. *Curso de Análise*, 7 ed., vol. 1. Rio de Janeiro: IMPA, 1976.

[60] Longoni, G., and Haghighat, A. Development of new quadrature sets with the "ordinate splitting" technique. In *Joint International Conference on Mathmatics and Computation* (2001).

[61] Madsen, N. K. Pointwise convergence of the three-dimensional discrete ordinate method. *SIAM Journal on Numerical Analysis v. 8*, n. 2 (1971), p.266–269.

[62] Miller, W. F., and Reed, W. H. Ray-effect mitigation methods for two-dimensional neutron transport theory. *Nuclear Science and Engineering v. 62*, n. 3 (1977), p.391–411.

[63] Morel, J. E., Wareing, T. A., Lowrie, R. B., and Parsons, D. K. Analysis of ray-effect mitigation techniques. *Nuclear Science and Engineering v. 144*, n. 1 (2003), p.1–22.

[64] Moura, F. W. D., Barichello, L. B., da Cunha, R. D., and Picoloto, C. B. On the influence of quadrature schemes for the iterative solution of linear systems in explicit Two-Dimensional discrete ordinates nodal formulations. In *PHYSOR* (2018).

[65] Nenê, J. S., and Barichello, L. B. *Raízes de polinômios de Legendre*: um problema de autovalores. Apresentado em 19 set. 2019. Pôster apresentado no XXXIX Congresso Nacional de Matemática Aplicada e Computacional de 16 à 20 de setembro de 2019, realizado em Uberlândia, MG, 2019.

[66] Rhoades, W., and Childs, R. *DORT/TORT two- and three-dimensional discrete ordinates transport, Version 2.7.3*. Oak Ridge: ORNL Radiation Shielding Information Center, 1993.

[67] Rodrigues, P. *Dinâmica de gases rarefeitos e transferência radiativa: aplicações em geometria cilíndrica*. Doutorado, Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Programa de Pós-Graduação em Engenharia Mecânica, 2003.

[68] Siewert, C. E. Poiseuille, thermal creep and Couette flow: results based on the CES model of the linearized Boltzmann equation. *European Journal of Mechanics - B/Fluids v. 21*, n. 5 (2002), p.579–597.

[69] Sjoden, G. E., and Haghighat, A. PENTRAN: A three-dimensional scalable transport code with complete phase-space decomposition. *Transactions of the American Nuclear Society v. 74* (1996), p.181–183.

[70] Spence, P. J. The generation of arbitrary order, non-classical, Gauss-type quadrature for transport applications. *Journal of Computational Physics v. 296* (2015), p.25–57.

[71] Stroud, A. H., and Secrest, D. *Gaussian Quadrature Formulas*. Englewood Cliffs: Prentice Hall, 1966.

[72] The MathWorks, Inc. Matlab. Release 2018b, 2018. Disponível em: https://www.mathworks.com/products/new_products/release2018b.html. Acesso em: 8 dez. 2019.

[73] Tres, A. *Análise de Esquemas de Aproximações para a Equação de Transporte Bidimensional em Ordenadas Discretas via Formulações Nodais*. Dou-

torado, Universidade Federal do Rio Grande do Sul. Instituto de Matemática. Programa de Pós-Graduação em Matemática Aplicada, 2015.

[74] Wick, G. C. Über ebene diffusions problem. *Z. Phys v. 120* (1943), p.702–705.

[75] Wilf, H. S. *Mathematics for the Physical Sciences*. New York: Dover Publications Inc, 1962.

APÊNDICE A: Códigos implementados em Fortran 95

```
SUBROUTINE abscissa_erf(k, h, a, b, x)
  Função responsável pelo cálculo dos nós da
  quadratura função erro utilizada no cálculo
  dos produtos internos

  Sintaxe:
  call abscissa_erf(k, h, a, b, x)

  Argumentos de entrada:
  k: valor inteiro; contador
  h: valor real; passo do processo iterativo
  a, b:  valores reais; limites de integração

  Argumentos de saída:
  x: valor real; valor do nó de quadratura

  Descrição:
  Gera o k-ésimo nó da quadratura função erro.

  Exemplos:
    real(tipo) :: h=0.004_tipo, a=0.0_tipo, b=1.0_tipo
    integer :: k = 10
    real(tipo) :: x
    call abscissa_erf(k, h, a, b, x)

SUBROUTINE abscissa_tanh_sinh(k, h, a, b, x)
  Função responsável pelo cálculo dos nós da
  quadratura tanh-sinh   utilizada no cálculo
  dos produtos internos

  Sintaxe:
  call abscissa_tanh_sinh(k, h, a, b, x)
```

```
  Argumentos de entrada:
  k: valor inteiro; contador
  h: valor real; passo do processo iterativo
  a, b:  valores reais; limites de integração

  Argumentos de saída:
  x: valor real; valor do nó de quadratura

  Descrição:
  Gera o k-ésimo nó da quadratura tanh-sinh.

  Exemplos:
    real(tipo) :: h=0.004_tipo, a=0.0_tipo, b=1.0_tipo
    integer :: k = 10
    real(tipo) :: x
    call abscissa_tanh_sinh(k, h, a, b, x)

SUBROUTINE weight_erf(k, h, a, b, w)
  Função responsável pelo cálculo dos pesos da
  quadratura função erro utilizada no cálculo dos
  produtos internos

  Sintaxe:
  call weight_erf(k, h, a, b, x)

  Argumentos de entrada:
  k: valor inteiro; contador
  h: valor real; passo do processo iterativo
  a, b:  valores reais; limites de integração

  Argumentos de saída:
  w: valor real; valor do peso de quadratura

  Descrição:
  Gera o k-ésimo peso da quadratura função erro.

  Exemplos:
    real(tipo) :: h=0.004_tipo, a=0.0_tipo, b=1.0_tipo
```

```
    integer :: k = 10
    real(tipo) :: w
    call weight_erf(k, h, a, b, w)

SUBROUTINE weight_tanh_sinh(k, h, a, b, w)
  Função responsável pelo cálculo dos pesos da
   quadratura tanh-sinh  utilizada no cálculo dos
   produtos internos

  Sintaxe:
  call weight_tanh_sinh(k, h, a, b, w)

  Argumentos de entrada:
  k: valor inteiro; contador
  h: valor real; passo do processo iterativo
  a, b:  valores reais; limites de integração

  Argumentos de saída:
  w: valor real; valor do peso de quadratura

  Descrição:
  Gera o k-ésimo peso da quadratura tanh-sinh.

  Exemplos:
    real(tipo) :: h=0.004_tipo, a=0.0_tipo, b=1.0_tipo
    integer :: k = 10
    real(tipo) :: x
    call weight_tanh_sinh(k, h, a, b, x)

SUBROUTINE fn_erfun_quad(a, b, h, tol, uk, wk, &
lgt, info)
  Gera os nós e pesos da quadratura função erro.

  Sintaxe:
  call fn_erfun_quad(a, b, h, tol, uk, wk, lgt, info)

  Utiliza as seguintes funções auxiliares:
  abscissa_erf
```

```
weight_erf

Argumentos de entrada:
a, b: valores reais; limites de integração
h: valor real; passo do processo iterativo
tol: valor real; tolerância da quadratura
(critério de parada)

Argumentos de saída:
uk: vetor real, lgt elementos; contém os nos da
quadratura função erro
wk: vetor real, lgt elementos; contém os pesos da
quadratura função erro
lgt: valor inteiro; quantidade de nos (e pesos)
info: valor inteiro; status da subrotina
  info = 0, subrotina conseguiu calcular os nos
  e pesos;
  info = -1, os vetores auxiliares uk1 e wk1 não
  puderam ser alocados;
  info = -2, os vetores uk e wk não puderam ser
  alocados;
  info = -3, o valor da tolerância não foi atingido.

Descrição:
Gera os nós e pesos da quadratura função erro com
nós gerados simetricamente em torno de zero.
Os nós são calculados através da subrotina
abscissa_erf e os pesos através da subrotina
weight_erf.
O processo de geração destes nós e pesos para quando
weight<tol, ou seja, quando o peso calculado para a
quadratura é pequeno o suficiente (menor que tol).
Os vetores de saída, uk e wk, são alocados dentro
dessa rotina, com tamanho lgt

Exemplos:
  real(tipo) :: a, b, h, tol
  real(tipo), dimension(:), allocatable :: uk, wk
```

```
    integer :: lgt, info
    a=0.0_tipo; b=1.0_tipo; h=0.004_tipo; tol=1e-16
    call fn_erfun_quad(a, b, h, tol, uk, wk, lgt, info)

SUBROUTINE fn_tanh_sinh_quad(a, b, h, tol, uk, wk, &
lgt, info)
  Gera os nós e pesos da quadratura tanh-sinh.

  Sintaxe:
  call fn_tanh_sinh_quad(a, b, h, tol, uk, wk, &
  lgt, info)

  Utiliza as seguintes funções auxiliares:
  abscissa_tanh_sinh
  weight_tanh_sinh

  Argumentos de entrada:
  a, b: valores reais; limites de integração
  h: valor real; passo do processo iterativo
  tol: valor real; tolerância da quadratura
  (critério de parada)

  Argumentos de saída:
  uk: vetor real, lgt elementos; contém os nos da
  quadratura tanh-sinh
  wk: vetor real, lgt elementos; contém os pesos da
  quadratura tanh-sinh
  lgt: valor inteiro; quantidade de nos (e pesos)
  info: valor inteiro; status da subrotina
    info = 0, subrotina conseguiu calcular os nos
    e pesos;
    info = -1, os vetores auxiliares uk1 e wk1 não
    puderam ser alocados;
    info = -2, os vetores uk e wk não puderam ser
    alocados;
    info = -3, o valor da tolerância não foi atingido.

  Descrição:
```

```
  Gera os nós e pesos da quadratura tanh-sinh com nós
  gerados simetricamente em torno de zero.
  Os nós são calculados através da subrotina
  abscissa_tanh_sinh e os pesos através da subrotina
  weight_tanh_sinh.
  O processo de geração destes nós e pesos para quando
  weight<tol, ou seja, quando o peso calculado para a
  quadratura é pequeno o suficiente (menor que tol).
  Os vetores de saída, uk e wk, são alocados dentro
  dessa rotina, com tamanho lgt.

  Exemplos:
    real(tipo) :: a, b, h, tol
    real(tipo), dimension(:), allocatable :: uk, wk
    integer :: lgt, info
    a=0.0_tipo, b=1.0_tipo, h=0.004_tipo, tol=1e-16
    call fn_qrs_full_tanh_sinh(a, b, h, tol, uk, wk, &
    lgt, info)

SUBROUTINE fn_orpolyval(x, n, an, bn, p0, p1, pn)
  Avalia o polinômio ortogonal de grau n (n > 1) em x.

  Sintaxe:
  call fn_orpolyval(x, n, an, bn, p0, p1, pn)

  Argumentos de entrada:
  x: vetor real, k elementos; contém os valores no
  qual o polinômio vai ser avaliado
  n: valor inteiro; grau do polinômio ortogonal (n > 1)
  an, bn: valor real; coeficientes da relação de
  recorrência de três termos que geram o polinômio
  ortogonal
  p0: vetor real, k elementos; polinômio ortogonal
  de grau n-2 avaliado em x
  p1: vetor real, k elementos; polinômio ortogonal de
  grau n-1 avaliado em x

  Argumentos de saída:
```

```
  pn: vetor real, k elementos; polinômio ortogonal de
  grau n avaliado em x

  Descrição:
  Avalia o polinômio ortogonal de grau n (n > 1) nas
  entradas do vetor x através da relação de recorrência
  de três termos que geram este polinômio.

  Exemplos:
    integer :: n=2, i
    real(tipo), dimension(k) :: p0, p1, pn, x
    real(tipo) :: an, bn
    an = 0.0_tipo; bn =1.0_tipo/3.0_tipo
    do i = 1, k
      x(i) = i
      p1(i) = i
    end do
    p0 = 1.0_tipo
    call fn_orpolyval(x, n, an, bn, p0, p1, pn)

SUBROUTINE fn_orpolyval_spence(x, n, an, bn, orpoly)
  Rotina proposta por Spence para avaliar o polinômio
  ortogonal de grau n em x

  Sintaxe:
  fn_orpolyval_spence(x, n, an, bn, orpoly)

  Argumentos de entrada:
  x: vetor real, k elementos; contém os valores no
  qual o polinômio vai ser avaliado
  n: valor inteiro; grau do polinômio ortogonal
  an, bn: vetor real, j elementos (j é a ordem da
  quadratura); coeficientes da relação de recorrência

  Argumentos de entrada e saída:
  orpoly: matriz real, (k, j) elementos; contém os
  polinômios ortogonais de graus 0, 1, 2, ..., n-1,
  avaliados em x, e na qual será salvo o polinômio
```

```
  Descrição:
  Transcrição da rotina proposta por Spence em matlab
  para a linguagem Fortran 95.
  Avalia o polinômio ortogonal de grau n nas entradas
  do vetor x, através da relação de recorrência de três
  termos que geram este polinômio.

  Exemplos:
    integer :: n=2, i
    real(tipo), dimension(k) :: x
    real(tipo), dimension(2) :: an, bn
    real(tipo), dimension(k, 2) :: orpoly
    an = 0.0_tipo; bn(1) = 0.0_tipo
    bn(2) = 1.0_tipo/3.0_tipo
    do i = 1, k
      x(i) = i
      orpoly(i, 2) = i
    end do
    orpoly(:, 1) = 1.0_tipo
    call fn_orpolyval_spence(x, n, an, bn, orpoly)

SUBROUTINE wfn(u, flag, wfnu)
  Avalia a função peso da quadratura no vetor u

  Sintaxe:
  call wfn(u, flag, wfnu)

  Argumentos de entrada:
  u: vetor real, k elementos; contém os valores no
  qual a função peso da quadratura será avaliada
  flag: valor inteiro; indica qual é a função peso
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 6: Azimutal_QRJ90
    OBS.: flag=5 não resolve problema de autovalor.
```

```
    Logo, não é necessário aqui.

  Argumentos de saída:
  wfnu: vetor real, k elementos; função peso da
  quadratura avaliada em u

  Exemplos:
    real(tipo), dimension(k) :: u, wfnu
    integer :: flag = 2
    u = 1.0_tipo
    call wfn(u, flag, wfnu)

SUBROUTINE calcula_pesos(n, nos, an, bn, iprod, pesos)
  Calcula os pesos da quadratura de gaussiana

  Sintaxe:
  call calcula_pesos(n, nos, an, bn, iprod, pesos)

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura gaussiana
  nos: vetor real, n elementos; contém as raízes do
  polinômio ortogonal de grau n
  an: vetor real, n elementos; contém os coeficientes
  relação de recorrência de 3 termos
  bn: vetor real, n-1 elementos; contém os coeficientes
  relação de recorrência de 3 termos
  iprod: vetor real, n elementos; contém o quadrado
  das normas dos polinômios ortogonais

  Argumentos de saída:
  pesos: vetore real, n elementos; contém os pesos da
  quadratura gaussiana

  Descrição:
  Calcula os pesos da quadratura gaussiana através
  de um somatório.
  A fórmula para o pesos é a equação (1.19) da
  referência.
```

```
  Exemplos:
    integer :: n = 2
    real(tipo), dimension(n) :: an, iprod, nos, pesos
    real(tipo), dimension(n-1) :: bn
    an = 0.0_tipo; bn = 1.0_tipo/3.0_tipo
    iprod(1) = 2.0_tipo; iprod(2) = 2.0_tipo/3.0_tipo
    nos(1) = -sqrt(3.0_tipo)/3.0_tipo
    nos(2) = -nos(1)
    call calcula_pesos(n, nos, an, bn, iprod, pesos)

SUBROUTINE fn_qrs_full_erf_slapack(xi, wi, n, flag, &
h, tol)
  Calcula os nós e pesos da quadratura QR via
  polinômios ortogonais em precisão simples

  Sintaxe:
  call fn_qrs_full_erf_slapack(xi, wi, n, flag, &
  h, tol)

  Utiliza as seguintes funções auxiliares:
  fn_tanh_sinh_quad
  ordena
  wfn
  fn_orpolyval
  calcula_pesos

  Argumentos de entrada:
  n: valor inteiro; grau do polinômio ortogonal
  h: valor real; passo da quadratura erf
  tol: valor real; tolerância quadratura erf
  flag: valor inteiro; determina qual quadratura será
  calculada
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
```

```
    flag = 6: Azimutal_QRJ90

  Argumentos de saída:
  xi: vetor real, n elementos; contém os nós da
  quadratura
  wi: vetor real, n elementos; contém os pesos da
  quadratura

  Descrição:
  Calcula os nós e pesos da quadratura gaussina de
  ordem n para precisão simples. Os nós são obtidos
  resolvendo um problema de autovalores para uma
  matriz tridiagonal simétrica, através da rotina
  SSTEVD da biblioteca LAPACK.
  Os pesos são calculados através de um somatório.
  Os coeficientes da relação de recorrência que geram
  os polinômios ortogonais são calculados através da
  quadratura função erro.

  Exemplos:
    integer :: n=2, flag=3
    real(tipo), dimension(n) :: xi, wi
    real(tipo) :: h=0.004_tipo, tol=1e-14
    call fn_qrs_full_erf_slapack(xi, wi, n, flag, &
    h, tol)

SUBROUTINE fn_qrs_full_erf_spence(xi, wi, n, flag)
  Calcula os nós e pesos da quadratura QR via
  polinômios ortogonais em precisão dupla de acordo
  com o algoritmo proposto por Spence.
  Utiliza os mesmos parâmetros e abordagem

  Sintaxe:
  call fn_qrs_full_erf_spence(xi, wi, n, flag)

  Utiliza as seguintes funções auxiliares:
  fn_erfun_quad
  ordena
```

```
wfn
fn_orpolyval_spence

Argumentos de entrada:
n: valor inteiro; grau do polinômio ortogonal
flag: valor inteiro; determina qual quadratura será
calculada
  flag = 1: Azimutal_QRS45
  flag = 2: Azimutal_QRA45
  flag = 3: Azimutal_QRS90
  flag = 4: Polar_QRA2D
  flag = 5: Azimutal_QRJ45
  flag = 6: Azimutal_QRJ90

Argumentos de saída:
xi: vetor real, n elementos; contém os nós da
quadratura
wi: vetor real, n elementos; contém os pesos da
quadratura

Descrição:
Transcrição da rotina proposta por Spence em matlab
para a linguagem Fortran 95.
Calcula os nós e pesos da quadratura gaussina de
ordem n para precisão dupla. Os nós são obtidos
resolvendo um problema de autovalores para uma matriz
tridiagonal simétrica e os pesos são calculados a
partir dos autovetores normalizados.
Os autovalores e autovetores são calculados através
da rotina DSTEVD da biblioteca LAPACK.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura erf.

Exemplos:
  integer :: n=2, flag=3
  real(tipo), dimension(n) :: xi, wi
  call fn_qrs_full_erf_spence(xi, wi, n, flag)
```

```
SUBROUTINE fn_qrs_full_erf(xi, wi, n, flag, h, tol)
  Calcula os nós e pesos da quadratura QR via
  polinômios ortogonais em precisão simples, dupla
  ou quádrupla

  Sintaxe:
  call fn_qrs_full_erf(xi, wi, n, flag, h, tol)

  Utiliza as seguintes funções auxiliares:
  fn_erfun_quad
  ordena
  wfn
  fn_orpolyval
  iter_qr
  calcula_pesos

  Argumentos de entrada:
  n: valor inteiro; grau do polinômio ortogonal
  h: valor real; passo da quadratura erf
  tol: valor real; tolerância quadratura erf
  flag: valor inteiro; determina qual quadratura será
  calculada
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90

  Argumentos de saída:
  xi: vetor real, n elementos; contém os nós da
  quadratura
  wi: vetor real, n elementos; contém os pesos da
  quadratura

  Descrição:
  Calcula os nós e pesos da quadratura gaussina de
```

```
  ordem n para precisão simples, dupla ou quádrupla.
  Os nós são obtidos resolvendo um problema de
  autovalores para uma matriz tridiagonal simétrica,
  através da rotina iter_qr, que foi implementada por
  nós. Os pesos são calculados através de um somatório.
  Os coeficientes da relação de recorrência que geram
  os polinômios ortogonais são calculados através da
  quadratura função erro.

  Exemplos:
    integer :: n=2, flag=3
    real(tipo), dimension(n) :: xi, wi
    real(tipo) :: h=0.004_tipo, tol=1e-14
    call fn_qrs_full_erf(xi, wi, n, flag, h, tol)

SUBROUTINE fn_qrs_full_tanh_sinh_slapack(xi, wi, n, &
flag, h, tol)
  Calcula os nós e pesos da quadratura QR via
  polinômios ortogonais em precisão simples

  Sintaxe:
  call fn_qrs_full_tanh_sinh_slapack(xi, wi, n, &
  flag, h, tol)

  Utiliza as seguintes funções auxiliares:
  fn_tanh_sinh_quad
  ordena
  wfn
  fn_orpolyval
  calcula_pesos

  Argumentos de entrada:
  n: valor inteiro; grau do polinômio ortogonal
  h: valor real; passo da quadratura tanh-sinh
  tol: valor real; tolerância quadratura tanh-sinh
  flag: valor inteiro; determina qual quadratura será
  calculada
    flag = 1: Azimutal_QRS45
```

```
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90

  Argumentos de saída:
  xi: vetor real, n elementos; contém os nós da
  quadratura
  wi: vetor real, n elementos; contém os pesos da
  quadratura

  Descrição:
  Calcula os nós e pesos da quadratura gaussina de
  ordem n para precisão dupla. Os nós são obtidos
  resolvendo um problema de autovalores para uma matriz
  tridiagonal simétrica, através da rotina SSTEVD da
  biblioteca LAPACK.
  Os pesos são calculados através de um somatório.
  Os coeficientes da relação de recorrência que geram
  os polinômios ortogonais são calculados através da
  quadratura tanh-sinh.

  Exemplos:
    integer :: n=2, flag=3
    real(tipo), dimension(n) :: xi, wi
    real(tipo) :: h=0.004_tipo, tol=1e/-14
    call fn_qrs_full_tanh_sinh_slapack(xi, wi, n, &
    flag, h, tol)

SUBROUTINE fn_qrs_full_tanh_sinh_dlapack(xi, wi, n, &
flag, h, tol)
  Calcula os nós e pesos da quadratura QR via
  polinômios ortogonais em precisão dupla

  Sintaxe:
  call fn_qrs_full_tanh_sinh_dlapack(xi, wi, n, &
  flag, h, tol)
```

```
Utiliza as seguintes funções auxiliares:
fn_tanh_sinh_quad
ordena
wfn
fn_orpolyval
calcula_pesos

Argumentos de entrada:
n: valor inteiro; grau do polinômio ortogonal
h: valor real; passo da quadratura tanh-sinh
tol: valor real; tolerância quadratura tanh-sinh
flag: valor inteiro; determina qual quadratura será
calculada
  flag = 1: Azimutal_QRS45
  flag = 2: Azimutal_QRA45
  flag = 3: Azimutal_QRS90
  flag = 4: Polar_QRA2D
  flag = 5: Azimutal_QRJ45
  flag = 6: Azimutal_QRJ90

Argumentos de saída:
xi: vetor real, n elementos; contém os nós da
quadratura
wi: vetor real, n elementos; contém os pesos da
quadratura

Descrição:
Calcula os nós e pesos da quadratura gaussina de
ordem n para precisão dupla. Os nós são obtidos
resolvendo um problema de autovalores para uma matriz
tridiagonal simétrica, através da rotina DSTEVD da
biblioteca LAPACK.
Os pesos são calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura tanh-sinh.
```

```
  Exemplos:
    integer :: n=2, flag=3
    real(tipo), dimension(n) :: xi, wi
    real(tipo) :: h=0.004_tipo, tol=1e-14
    call fn_qrs_full_tanh_sinh_dlapack(xi, wi, n, &
    flag, h, tol)

SUBROUTINE fn_qrs_full_tanh_sinh_spence(xi, wi, n, flag)
  Calcula os nós e pesos da quadratura QR via
  polinômios ortogonais em precisão dupla baseada no
  algoritmo proposto por Spence.
  Utiliza os mesmos parâmetros e abordagem

  Sintaxe:
  call fn_qrs_full_tanh_sinh_spence(xi, wi, n, flag)

  Utiliza as seguintes funções auxiliares:
  fn_tanh_sinh_quad
  ordena
  wfn
  fn_orpolyval_spence

  Argumentos de entrada:
  n: valor inteiro; grau do polinômio ortogonal
  flag: valor inteiro; determina qual quadratura será
  calculada
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90

  Argumentos de saída:
  xi: vetor real, n elementos; contém os nós da
  quadratura
  wi: vetor real, n elementos; contém os pesos da
  quadratura
```

```
  Descrição:
  Calcula os nós e pesos da quadratura gaussina de
  ordem n para precisão dupla. Os nós são obtidos
  resolvendo um problema de autovalores para uma
  matriz tridiagonal simétrica e os pesos são
  calculados a partir dos autovetores normalizados.
  Os autovalores e autovetores são calculados através
  da rotina DSTEVD da biblioteca LAPACK.
  Os coeficientes da relação de recorrência que geram
  os polinômios ortogonais são calculados através da
  quadratura tanh-sinh.

  OBS.:
  A única diferença desta rotina para a rotina
  fn_qrs_full_erf_spence está na quadratura utilizada
  para gerar os polinômios ortogonais.

  Exemplos:
    integer :: n=2, flag=3
    real(tipo), dimension(n) :: xi, wi
    call fn_qrs_full_tanh_sinh_spence(xi, wi, n, flag)

SUBROUTINE fn_qrs_full_tanh_sinh(xi, wi, n, flag, &
h, tol)
  Calcula os nós e pesos da quadratura QR via
  polinômios ortogonais em precisão simples, dupla
  ou quádrupla

  Sintaxe:
  call fn_qrs_full_tanh_sinh(xi, wi, n, flag, h, tol)

  Utiliza as seguintes funções auxiliares:
  fn_tanh_sinh_quad
  ordena
  wfn
  fn_orpolyval
  iter_qr
```

```
calcula_pesos

Argumentos de entrada:
n: valor inteiro; grau do polinômio ortogonal
h: valor real; passo da quadratura tanh-sinh
tol: valor real; tolerância quadratura tanh-sinh
flag: valor inteiro; determina qual quadratura será
calculada
  flag = 1: Azimutal_QRS45
  flag = 2: Azimutal_QRA45
  flag = 3: Azimutal_QRS90
  flag = 4: Polar_QRA2D
  flag = 5: Azimutal_QRJ45
  flag = 6: Azimutal_QRJ90

Argumentos de saída:
xi: vetor real, n elementos; contém os nós da
quadratura
wi: vetor real, n elementos; contém os pesos da
quadratura

Descrição:
Calcula os nós e pesos da quadratura gaussina de
ordem n para precisão simples, dupla ou quádrupla.
Os nós são obtidos resolvendo um problema de
autovalores para uma matriz tridiagonal simétrica,
através da rotina iter_qr, que foi implementada
por nós.
Os pesos são calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura tanh-sinh.

Exemplos:
  integer :: n=2, flag=3
  real(tipo), dimension(n) :: xi, wi
  real(tipo) :: h=0.004_tipo, tol=1e-14
  call fn_qrs_full_tanh_sinh(xi, wi, n, flag, h, tol)
```

```
SUBROUTINE quad_qr_r_erf_slapack(mu, eta, xi, pesos, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR retangular para o hemisfério
  superior da esfera unitária em precisão simples

  Sintaxe:
  call quad_qr_r_erf_slapack(mu, eta, xi, pesos, &
  ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_erf_slapack

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: valor inteiro; determina a ordem da
  quadratura azimutal
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: valor inteiro; determina a flag da
  quadratura azimutal a ser utilizada
   flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90
  h: valor real; passo da quadratura função erro
  tol: valor real; tolerância da quadratura função erro

  Argumentos de saída:
  mu: vetor real, 4*ordem_azi*ordem_polar elementos;
  armazena a direção mu
  eta: vetor real, 4*ordem_azi*ordem_polar elementos;
  armazena a direção eta
  xi: vetor real, 4*ordem_azi*ordem_polar elementos;
```

```
armazena a direção xi
pesos: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura QR
retangular em precisão simples para o hemisfério
superior da esfera unitária.
Cada vetor tem tamanho 4*ordem_azi*ordem_polar, sendo
que o número de direções discretas por octante da
esfera unitária é M = ordem_azi*ordem_polar.
É utilizada a mesma ordem azimutal em todos os níveis
polares.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina SSTEVD da
biblioteca LAPACK, e os respectivos pesos são
calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura função erro.


Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol
  integer :: flag_polar, flag_azi, ordem_polar, &
  ordem_azi, M
  h = 0.004_tipo; tol = 1e-16
  flag_polar = 4; flag_azi = 2
  ordem_polar = 5; ordem_azi = 3*ordem_polar
  M = ordem_polar*ordem_azi
  allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
  call quad_qr_r_erf_slapack(mu, eta, xi, pesos, &
```

```
    ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

SUBROUTINE quad_qr_r_erf_dlapack(mu, eta, xi, pesos, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR retangular para o hemisfério
  superior da esfera unitária em precisão dupla

  Sintaxe:
  call quad_qr_r_erf_dlapack(mu, eta, xi, pesos, &
  ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_erf_dlapack

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: valor inteiro; determina a ordem da
  quadratura azimutal
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: valor inteiro; determina a flag da
  quadratura azimutal a ser utilizada
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90
  h: valor real; passo da quadratura função erro
  tol: valor real; tolerância da quadratura função erro

  Argumentos de saída:
  mu: vetor real, 4*ordem_azi*ordem_polar elementos;
  armazena a direção mu
  eta: vetor real, 4*ordem_azi*ordem_polar elementos;
  armazena a direção eta
```

```
xi: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção xi
pesos: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura QR
retangular em precisão dupla para o hemisfério
superior da esfera unitária.
Cada vetor tem tamanho 4*ordem_azi*ordem_polar,
sendo que o número de direções discretas por octante
da esfera unitária é M = ordem_azi*ordem_polar.
É utilizada a mesma ordem azimutal em todos os
níveis polares.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina DSTEVD da
biblioteca LAPACK, e os respectivos pesos são
calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura função erro.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol
  integer :: flag_polar, flag_azi, ordem_polar, &
  ordem_azi, M
  h = 0.004_tipo; tol = 1e-16
  flag_polar = 4; flag_azi = 2
  ordem_polar = 5; ordem_azi = 3*ordem_polar
  M = ordem_polar*ordem_azi
  allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
  call quad_qr_r_erf_dlapack(mu, eta, xi, pesos, &
```

```
    ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

SUBROUTINE quad_qr_r_erf(mu, eta, xi, pesos, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR retangular para o hemisfério
  superior da esfera unitária em precisão simples,
  dupla ou quádrupla

  Sintaxe:
  call quad_qr_r_erf(mu, eta, xi, pesos, ordem_polar, &
  ordem_azi, flag_polar, flag_azi, h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_erf

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: valor inteiro; determina a ordem da
  quadratura azimutal
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: valor inteiro; determina a flag
  da quadratura azimutal a ser utilizada
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90
  h: valor real; passo da quadratura função erro
  tol: valor real; tolerância da quadratura função erro

  Argumentos de saída:
  mu: vetor real, 4*ordem_azi*ordem_polar elementos;
  armazena a direção mu
  eta: vetor real, 4*ordem_azi*ordem_polar elementos;
```

```
armazena a direção eta
xi: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção xi
pesos: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
QR retangular em precisão simples, dupla ou
quádrupla para o hemisfério superior da esfera
unitária.
Cada vetor tem tamanho 4*ordem_azi*ordem_polar,
sendo que o número de direções discretas por octante
da esfera unitária é M = ordem_azi*ordem_polar.
É utilizada a mesma ordem azimutal em todos os níveis
polares.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina iter_qr, e
os respectivos pesos são calculados através de um
somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura função erro.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol
  integer :: flag_polar, flag_azi, ordem_polar, &
  ordem_azi, M
  h = 0.004_tipo; tol = 1e-16
  flag_polar = 4; flag_azi = 2
  ordem_polar = 5; ordem_azi = 3*ordem_polar
  M = ordem_polar*ordem_azi
```

```
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_r_erf(mu, eta, xi, pesos, &
    ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

SUBROUTINE quad_qr_r_tanh_sinh_slapack(mu, eta, xi, &
pesos, ordem_polar, ordem_azi, flag_polar, flag_azi, &
h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR retangular para o hemisfério
  superior da esfera unitária em precisão simples

  Sintaxe:
  call quad_qr_r_tanh_sinh_slapack(mu, eta, xi, &
  pesos, ordem_polar, ordem_azi, flag_polar, &
  flag_azi, h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_tanh_sinh_slapack

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: valor inteiro; determina a ordem da
  quadratura azimutal
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: valor inteiro; determina a flag da
  quadratura azimutal a ser utilizada
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90
  h: valor real; passo da quadratura tanh-sinh
  tol: valor real; tolerância da quadratura tanh-sinh

  Argumentos de saída:
```

```
mu: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção mu
eta: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção eta
xi: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção xi
pesos: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
QR retangular em precisão simples para o hemisfério
superior da esfera unitária.
Cada vetor tem tamanho 4*ordem_azi*ordem_polar, sendo
que o número de direções discretas por octante da
esfera unitária é M = ordem_azi*ordem_polar.
É utilizada a mesma ordem azimutal em todos os níveis
polares.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina SSTEVD da
biblioteca LAPACK, e os respectivos pesos são
calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura tanh-sinh.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol
  integer :: flag_polar, flag_azi, ordem_polar, &
  ordem_azi, M
  h = 0.004_tipo; tol = 1e-16
  flag_polar = 4; flag_azi = 2
```

```
    ordem_polar = 5; ordem_azi = 3*ordem_polar
    M = ordem_polar*ordem_azi
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_r_tanh_sinh_slapack(mu, eta, xi, &
    pesos, ordem_polar, ordem_azi, flag_polar, &
    flag_azi, h, tol)

SUBROUTINE quad_qr_r_tanh_sinh_dlapack(mu, eta, xi, &
pesos, ordem_polar, ordem_azi, flag_polar, flag_azi, &
h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR retangular para o hemisfério
  superior da esfera unitária em precisão dupla

  Sintaxe:
  call quad_qr_r_tanh_sinh_dlapack(mu, eta, xi, &
  pesos, ordem_polar, ordem_azi, flag_polar, flag_azi, &
  h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_tanh_sinh_dlapack

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: valor inteiro; determina a ordem da
  quadratura azimutal
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: valor inteiro; determina a flag da
  quadratura azimutal a ser utilizada
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90
  h: valor real; passo da quadratura tanh-sinh
```

```
tol: valor real; tolerância da quadratura tanh-sinh

Argumentos de saída:
mu: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção mu
eta: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção eta
xi: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção xi
pesos: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura QR
retangular em precisão dupla para o hemisfério
superior da esfera unitária.
Cada vetor tem tamanho 4*ordem_azi*ordem_polar, sendo
que o número de direções discretas por octante da
esfera unitária é M = ordem_azi*ordem_polar.
É utilizada a mesma ordem azimutal em todos os níveis
polares.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina DSTEVD da
biblioteca LAPACK, e os respectivos pesos são
calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura tanh-sinh.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol
  integer :: flag_polar, flag_azi, ordem_polar, &
```

```
    ordem_azi, M
    h = 0.004_tipo; tol = 1e-16
    flag_polar = 4; flag_azi = 2
    ordem_polar = 5; ordem_azi = 3*ordem_polar
    M = ordem_polar*ordem_azi
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_r_tanh_sinh_dlapack(mu, eta, xi, &
    pesos, ordem_polar, ordem_azi, flag_polar, &
    flag_azi, h, tol)

SUBROUTINE quad_qr_r_tanh_sinh(mu, eta, xi, pesos, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)
  Gera as direções de quadratura e os respectivos pesos
  da quadratura QR retangular para o hemisfério superior
  da esfera unitária em precisão simples, dupla ou
  quádrupla

  Sintaxe:
  call quad_qr_r_tanh_sinh(mu, eta, xi, pesos, &
  ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_tanh_sinh

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: valor inteiro; determina a ordem da
  quadratura azimutal
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: valor inteiro; determina a flag da
  quadratura azimutal a ser utilizada
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
```

```
  flag = 6: Azimutal_QRJ90
h: valor real; passo da quadratura tanh-sinh
tol: valor real; tolerância da quadratura tanh-sinh

Argumentos de saída:
mu: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção mu
eta: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção eta
xi: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena a direção xi
pesos: vetor real, 4*ordem_azi*ordem_polar elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura QR
retangular em precisão simples, dupla ou quádrupla
para o hemisfério superior da esfera unitária.
Cada vetor tem tamanho 4*ordem_azi*ordem_polar, sendo
que o número de direções discretas por octante da
esfera unitária é M = ordem_azi*ordem_polar.
É utilizada a mesma ordem azimutal em todos os níveis
polares.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina iter_qr, e
os respectivos pesos são calculados através de um
somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura tanh-sinh.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
```

```
    real(tipo):: h, tol
    integer :: flag_polar, flag_azi, ordem_polar, &
    ordem_azi, M
    h = 0.004_tipo; tol = 1e-16
    flag_polar = 4; flag_azi = 2
    ordem_polar = 5; ordem_azi = 3*ordem_polar
    M = ordem_polar*ordem_azi
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_r_tanh_sinh(mu, eta, xi, pesos, &
    ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

SUBROUTINE quad_qr_t_erf_slapack(mu, eta, xi, pesos, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR triangular para o hemisfério
  superior da esfera unitária em precisão simples

  Sintaxe:
  call quad_qr_t_erf_slapack(mu, eta, pesos, &
  ordem_polar, ordem_azi, flag_polar, flag_azi, &
  h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_erf_slapack

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: vetor inteiro, ordem_polar elementos;
  determina a ordem da quadratura azimutal utilizada
  em cada nível polar. A entrada i do vetor contém a
  ordem azimutal do nível polar i
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: vetor inteiro, ordem_polar elementos;
  determina a flag da quadratura azimutal utilizada em
  cada nível polar. A entrada i do vetor contém a
  flag azimutal do nível polar i
```

```
  flag = 1: Azimutal_QRS45
  flag = 2: Azimutal_QRA45
  flag = 3: Azimutal_QRS90
  flag = 4: Polar_QRA2D
  flag = 5: Azimutal_QRJ45
  flag = 6: Azimutal_QRJ90
h: valor real; passo da quadratura função erro
tol: valor real; tolerância da quadratura função erro

Argumentos de saída:
mu: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção mu
eta: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção eta
xi: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção xi
pesos: vetor real, 4*sum(ordem_azi) elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura QR
triangular em precisão simples para o hemisfério
superior da esfera unitária.
Cada vetor tem tamanho 4*sum(ordem_azi), sendo que o
número de direções discretas por octante da esfera
unitária é M = sum(ordem_azi) (sum(ordem_azi)
representa o somatório das entradas do vetor
ordem_azi).
O vetor ordem_azimutal determina a ordem azimutal
utilizada em cada nível polar, de modo que, a
i-ésima posição do vetor contém a ordem azimutal do
i-ésimo nível polar (o 1º nível polar é o mais
próximo da borda da esfera e o último nível polar
é o mais próximo do centro da esfera).
De modo análogo, o vetor flag_azimutal determina a
flag azimutal utilizada em cada nível polar, de modo
que, a i-ésima posição do vetor contém a flag
azimutal do i-ésimo nível polar.
```

```
  Temos theta \in [0, pi] e phi \in [0, 2pi], em que
  theta é o ângulo polar medido a partir do eixo xi e
  phi o ângulo azimutal medido a partir do eixo x.
  A soma dos pesos de quadratura é 4*pi.
  O problema de autovalores para as quadraturas
  gaussianas é resolvido através da rotina SSTEVD da
  biblioteca LAPACK, e os respectivos pesos são
  calculados através de um somatório.
  Os coeficientes da relação de recorrência que geram
  os polinômios ortogonais são calculados através da
  quadratura função erro.

  Exemplos:
    real(tipo), dimension(:), allocatable :: mu, eta, &
    xi, pesos
    real(tipo):: h, tol
    integer :: flag_polar, ordem_azi, M, i
    integer, dimension(:), allocatable :: flag_azi, &
    ordem_azi
    h = 0.004_tipo; tol = 1e-16
    flag_polar = 4; ordem_polar = 5
    allocate(flag_azi(ordem_polar), &
    ordem_azi(ordem_polar))
    do i = 1, ordem_polar
      ordem_azi(i) = i
    end do
    flag_azi = 3
    M = sum(ordem_azi)
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_t_erf_slapack(mu, eta, xi, pesos, &
    ordem_polar, ordem_azi, flag_polar, flag_azi, &
    h, tol)

SUBROUTINE quad_qr_t_erf_dlapack(mu, eta, xi, pesos, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)
  Gera as direções de quadratura e os respectivos pesos
  da quadratura QR triangular para o hemisfério superior
  da esfera unitária em precisão dupla
```

```
Sintaxe:
call quad_qr_t_erf_dlapack(mu, eta, xi, pesos, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

Utiliza a seguinte função auxiliar:
fn_qrs_full_erf_dlapack

Argumentos de entrada:
ordem_polar: valor inteiro; determina a ordem da
quadratura polar
ordem_azi: vetor inteiro, ordem_polar elementos;
determina a ordem da quadratura azimutal utilizada
em cada nível polar. A entrada i do vetor contém a
ordem azimutal do nível polar i
flag_polar: valor inteiro; determina a flag da
quadratura polar
flag_azimutal: vetor inteiro, ordem_polar elementos;
determina a flag da quadratura azimutal utilizada em
cada nível polar. A entrada i do vetor contém a flag
azimutal do nível polar i
  flag = 1: Azimutal_QRS45
  flag = 2: Azimutal_QRA45
  flag = 3: Azimutal_QRS90
  flag = 4: Polar_QRA2D
  flag = 5: Azimutal_QRJ45
  flag = 6: Azimutal_QRJ90
h: valor real; passo da quadratura função erro
tol: valor real; tolerância da quadratura função erro

Argumentos de saída:
mu: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção mu
eta: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção eta
xi: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção xi
pesos: vetor real, 4*sum(ordem_azi) elementos;
```

```
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura QR
triangular em precisão dupla para o hemisfério
superior da esfera unitária.
Cada vetor tem tamanho 4*sum(ordem_azi), sendo que o
número de direções discretas por octante da esfera
unitária é M = sum(ordem_azi) (sum(ordem_azi)
representa o somatório das entradas do vetor
ordem_azi).
O vetor ordem_azimutal determina a ordem azimutal
utilizada em cada nível polar, de modo que, a i-ésima
posição do vetor contém a ordem azimutal do i-ésimo
nível polar (o 1° nível polar é o mais próximo da
borda da esfera e o último nível polar é o mais
próximo do centro da esfera).
De modo análogo, o vetor flag_azimutal determina a
flag azimutal utilizada em cada nível polar, de modo
que, a i-ésima posição do vetor contém a flag
azimutal do i-ésimo nível polar.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina DSTEVD da
biblioteca LAPACK, e os respectivos pesos são
calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura função erro.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol
  integer :: flag_polar, ordem_azi, M, i
```

```
    integer, dimension(:), allocatable :: flag_azi, &
    ordem_azi
    h = 0.004_tipo; tol = 1e-16
    flag_polar = 4; ordem_polar = 5
    allocate(flag_azi(ordem_polar), &
    ordem_azi(ordem_polar))
    do i = 1, ordem_polar
      ordem_azi(i) = i
    end do
    flag_azi = 3
    M = sum(ordem_azi)
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_t_erf_dlapack(mu, eta, xi, pesos, &
    ordem_polar, ordem_azi, flag_polar, flag_azi, &
    h, tol)

SUBROUTINE quad_qr_t_erf(mu, eta, pesos, xi, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR triangular para o hemisfério
  superior da esfera unitária em precisão simples,
  dupla ou quádrupla

  Sintaxe:
  call quad_qr_t_erf(mu, eta, pesos, xi, ordem_polar, &
  ordem_azi, flag_polar, flag_azi, h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_erf

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: vetor inteiro, ordem_polar elementos;
  determina a ordem da quadratura azimutal utilizada
  em cada nível polar. A entrada i do vetor contém a
  ordem azimutal do nível polar i
  flag_polar: valor inteiro; determina a flag da
```

```
quadratura polar
flag_azimutal: vetor inteiro, ordem_polar elementos;
determina a flag da quadratura azimutal utilizada em
cada nível polar. A entrada i do vetor contém a flag
azimutal do nível polar i
  flag = 1: Azimutal_QRS45
  flag = 2: Azimutal_QRA45
  flag = 3: Azimutal_QRS90
  flag = 4: Polar_QRA2D
  flag = 5: Azimutal_QRJ45
  flag = 6: Azimutal_QRJ90
h: valor real; passo da quadratura função erro
tol: valor real; tolerância da quadratura função erro

Argumentos de saída:
mu: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção mu
eta: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção eta
xi: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção xi
pesos: vetor real, 4*sum(ordem_azi) elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
QR triangular em precisão dupla para o hemisfério
superior da esfera unitária.
Cada vetor tem tamanho 4*sum(ordem_azi), sendo que
o número de direções discretas por octante da esfera
unitária é M = sum(ordem_azi) (sum(ordem_azi)
representa o somatório das entradas do vetor
ordem_azi).
O vetor ordem_azimutal determina a ordem azimutal
utilizada em cada nível polar, de modo que, a i-ésima
posição do vetor contém a ordem azimutal do i-ésimo
nível polar (o 1º nível polar é o mais próximo da
borda da esfera e o último nível polar é o mais
```

```
próximo do centro da esfera).
De modo análogo, o vetor flag_azimutal determina a
flag azimutal utilizada em cada nível polar, de modo
que, a i-ésima posição do vetor contém a flag
azimutal do i-ésimo nível polar.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina iter_qr, e
os respectivos pesos são calculados através de um
somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura função erro.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol
  integer :: flag_polar, ordem_azi, M, i
  integer, dimension(:), allocatable :: flag_azi, &
  ordem_azi
  h = 0.004_tipo; tol = 1e-16
  flag_polar = 4; ordem_polar = 5
  allocate(flag_azi(ordem_polar), &
  ordem_azi(ordem_polar))
  do i = 1, ordem_polar
    ordem_azi(i) = i
  end do
  flag_azi = 3
  M = sum(ordem_azi)
  allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
  call quad_qr_t_erf(mu, eta, pesos, xi, &
  ordem_polar, ordem_azi, flag_polar, flag_azi, &
  h, tol)
```

```
SUBROUTINE quad_qr_t_tanh_sinh_slapack(mu, eta, xi, &
pesos, ordem_polar, ordem_azi, flag_polar, flag_azi, &
h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR triangular para o
  hemisfério superior da esfera unitária em
  precisão simples

  Sintaxe:
  call quad_qr_t_tanh_sinh_slapack(mu, eta, xi, &
  pesos, ordem_polar, ordem_azi, flag_polar, flag_azi, &
  h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_tanh_sinh_slapack

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: vetor inteiro, ordem_polar elementos;
  determina a ordem da quadratura azimutal utilizada
  em cada nível polar. A entrada i do vetor contém a
  ordem azimutal do nível polar i
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: vetor inteiro, ordem_polar elementos;
  determina a flag da quadratura azimutal utilizada em
  cada nível polar.
  A entrada i do vetor contém a flag azimutal do nível
  polar i
    flag = 1: Azimutal_QRS45
    flag = 2: Azimutal_QRA45
    flag = 3: Azimutal_QRS90
    flag = 4: Polar_QRA2D
    flag = 5: Azimutal_QRJ45
    flag = 6: Azimutal_QRJ90
  h: valor real; passo da quadratura função erro
  tol: valor real; tolerância da quadratura função erro
```

```
Argumentos de saída:
mu: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção mu
eta: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção eta
xi: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção xi
pesos: vetor real, 4*sum(ordem_azi) elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
QR triangular em precisão simples para o hemisfério
superior da esfera unitária.
Cada vetor tem tamanho 4*sum(ordem_azi), sendo que o
número de direções discretas por octante da esfera
unitária é M = sum(ordem_azi) (sum(ordem_azi)
representa o somatório das entradas do vetor
ordem_azi).
O vetor ordem_azimutal determina a ordem azimutal
utilizada em cada nível polar, de modo que, a i-ésima
posição do vetor contém a ordem azimutal do i-ésimo
nível polar (o 1º nível polar é o mais próximo da
borda da esfera e o último nível polar é o mais
próximo do centro da esfera).
De modo análogo, o vetor flag_azimutal determina a
flag azimutal utilizada em cada nível polar, de modo
que, a i-ésima posição do vetor contém a flag
azimutal do i-ésimo nível polar.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina SSTEVD da
biblioteca LAPACK, e os respectivos pesos são
calculados através de um somatório.
```

```
  Os coeficientes da relação de recorrência que geram
  os polinômios ortogonais são calculados através da
  quadratura tanh-sinh.

  Exemplos:
    real(tipo), dimension(:), allocatable :: mu, eta, &
    xi, pesos
    real(tipo):: h, tol
    integer :: flag_polar, ordem_azi, M, i
    integer, dimension(:), allocatable :: flag_azi, &
    ordem_azi
    h = 0.004_tipo; tol = 1e-16
    flag_polar = 4; ordem_polar = 5
    allocate(flag_azi(ordem_polar), &
    ordem_azi(ordem_polar))
    do i = 1, ordem_polar
      ordem_azi(i) = i
    end do
    flag_azi = 3
    M = sum(ordem_azi)
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_t_tanh_sinh_slapack(mu, eta, xi, &
    pesos, ordem_polar, ordem_azi, flag_polar, &
    flag_azi, h, tol)

SUBROUTINE quad_qr_t_tanh_sinh_dlapack(mu, eta, xi, &
pesos, ordem_polar, ordem_azi, flag_polar, flag_azi, &
h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR triangular para o hemisfério
  superior da esfera unitária em precisão dupla

  Sintaxe:
  call quad_qr_t_tanh_sinh_dlapack(mu, eta, xi, &
  pesos, ordem_polar, ordem_azi, flag_polar, &
  flag_azi, h, tol)

  Utiliza a seguinte função auxiliar:
```

```
fn_qrs_full_tanh_sinh_dlapack

Argumentos de entrada:
ordem_polar: valor inteiro; determina a ordem da
quadratura polar
ordem_azi: vetor inteiro, ordem_polar elementos;
determina a ordem da quadratura azimutal utilizada
em cada nível polar. A entrada i do vetor contém a
ordem azimutal do nível polar i
flag_polar: valor inteiro; determina a flag da
quadratura polar
flag_azimutal: vetor inteiro, ordem_polar elementos;
determina a flag da quadratura azimutal utilizada em
cada nível polar.
A entrada i do vetor contém a flag azimutal do nível
polar i
  flag = 1: Azimutal_QRS45
  flag = 2: Azimutal_QRA45
  flag = 3: Azimutal_QRS90
  flag = 4: Polar_QRA2D
  flag = 5: Azimutal_QRJ45
  flag = 6: Azimutal_QRJ90
h: valor real; passo da quadratura função erro
tol: valor real; tolerância da quadratura função erro

Argumentos de saída:
mu: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção mu
eta: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção eta
xi: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção xi
pesos: vetor real, 4*sum(ordem_azi) elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
QR triangular em precisão dupla para o hemisfério
```

```
superior da esfera unitária.
Cada vetor tem tamanho 4*sum(ordem_azi), sendo que
o número de direções discretas por octante da esfera
unitária é M = sum(ordem_azi) (sum(ordem_azi)
representa o somatório das entradas do vetor
ordem_azi).
O vetor ordem_azimutal determina a ordem azimutal
utilizada em cada nível polar, de modo que, a i-ésima
posição do vetor contém a ordem azimutal do i-ésimo
nível polar (o 1° nível polar é o mais próximo da
borda da esfera e o último nível polar é o mais
próximo do centro da esfera).
De modo análogo, o vetor flag_azimutal determina a
flag azimutal utilizada em cada nível polar, de modo
que, a i-ésima posição do vetor contém a flag
azimutal do i-ésimo nível polar.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para as quadraturas
gaussianas é resolvido através da rotina DSTEVD da
biblioteca LAPACK, e os respectivos pesos são
calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura tanh-sinh.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol
  integer :: flag_polar, ordem_azi, M, i
  integer, dimension(:), allocatable :: flag_azi, &
  ordem_azi
  h = 0.004_tipo; tol = 1e-16
  flag_polar = 4; ordem_polar = 5
  allocate(flag_azi(ordem_polar), &
```

```
    ordem_azi(ordem_polar))
    do i = 1, ordem_polar
      ordem_azi(i) = i
    end do
    flag_azi = 3
    M = sum(ordem_azi)
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_t_tanh_sinh_dlapack(mu, eta, xi, &
    pesos, ordem_polar, ordem_azi, flag_polar, &
    flag_azi, h, tol)

SUBROUTINE quad_qr_t_tanh_sinh(mu, eta, xi, pesos, &
ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR triangular para o hemisfério
  superior da esfera unitária em precisão simples,
  dupla ou quádrupla

  Sintaxe:
  callquad_qr_t_tanh_sinh(mu, eta, xi, pesos, &
  ordem_polar, ordem_azi, flag_polar, flag_azi, h, tol)

  Utiliza a seguinte função auxiliar:
  fn_qrs_full_tanh_sinh

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi: vetor inteiro, ordem_polar elementos;
  determina a ordem da quadratura azimutal utilizada
  em cada nível polar. A entrada i do vetor contém a
  ordem azimutal do nível polar i
  flag_polar: valor inteiro; determina a flag da
  quadratura polar
  flag_azimutal: vetor inteiro, ordem_polar elementos;
  determina a flag da quadratura azimutal utilizada em
  cada nível polar. A entrada i do vetor contém a flag
  azimutal do nível polar i
```

```
  flag = 1: Azimutal_QRS45
  flag = 2: Azimutal_QRA45
  flag = 3: Azimutal_QRS90
  flag = 4: Polar_QRA2D
  flag = 5: Azimutal_QRJ45
  flag = 6: Azimutal_QRJ90
h: valor real; passo da quadratura função erro
tol: valor real; tolerância da quadratura função erro

Argumentos de saída:
mu: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção mu
eta: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção eta
xi: vetor real, 4*sum(ordem_azi) elementos;
armazena a direção xi
pesos: vetor real, 4*sum(ordem_azi) elementos;
armazena os pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
QR triangular em precisão simples, dupla ou
quádrupla para o hemisfério superior da esfera
unitária.
Cada vetor tem tamanho 4*sum(ordem_azi), sendo que
o número de direções discretas por octante da esfera
unitária é M = sum(ordem_azi) (sum(ordem_azi)
representa o somatório das entradas do vetor
ordem_azi).
O vetor ordem_azimutal determina a ordem azimutal
utilizada em cada nível polar, de modo que, a i-ésima
posição do vetor contém a ordem azimutal do i-ésimo
nível polar (o 1º nível polar é o mais próximo da
borda da esfera e o último nível polar é o mais
próximo do centro da esfera).
De modo análogo, o vetor flag_azimutal determina a
flag azimutal utilizada em cada nível polar, de modo
que, a i-ésima posição do vetor contém a flag
```

```
  azimutal do i-ésimo nível polar.
  Temos theta \in [0, pi] e phi \in [0, 2pi], em que
  theta é o ângulo polar medido a partir do eixo xi e
  phi o ângulo azimutal medido a partir do eixo x.
  A soma dos pesos de quadratura é 4*pi.
  O problema de autovalores para as quadraturas
  gaussianas é resolvido através da rotina iter_qr, e
  os respectivos pesos são calculados através de um
  somatório.
  Os coeficientes da relação de recorrência que geram
  os polinômios ortogonais são calculados através da
  quadratura tanh-sinh.

  Exemplos:
    real(tipo), dimension(:), allocatable :: mu, eta, &
    xi, pesos
    real(tipo):: h, tol
    integer :: flag_polar, ordem_azi, M, i
    integer, dimension(:), allocatable :: flag_azi, &
    ordem_azi
    h = 0.004_tipo; tol = 1e-16
    flag_polar = 4; ordem_polar = 5
    allocate(flag_azi(ordem_polar), &
    ordem_azi(ordem_polar))
    do i = 1, ordem_polar
      ordem_azi(i) = i
    end do
    flag_azi = 3
    M = sum(ordem_azi)
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_t_tanh_sinh(mu, eta, xi, pesos, &
    ordem_polar, ordem_azi, flag_polar, flag_azi, &
    h, tol)

SUBROUTINE cria_matriz(n, diag_pri, diag_inf)
  Cria a matriz tridiagonal simétrica utilizada no
  cálculo dos nós e pesos da quadratura de
  Gauss-Legendre
```

```
  Sintaxe:
  call cria_matriz(n, diag_pri, diag_inf)

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss_Legendre. A ordem deve ser um natural par.

  Argumentos de saída:
  diag_pri: vetor real, n/2 elementos; contém os
  elementos da diagonal principal
  diag_inf: vetor real, n/2-1 elementos; contém os
  elementos da diagonal inferior

  Descrição:
  Cria matriz simétrica tridiagonal utilizada no
  cálculo dos nós e pesos da quadratura de
  Gauss-Legendre. A matriz é armazenada como dois
  vetores, um contendo a diagonal principal e outro
  a subdiagonal.

  Exemplos:
    integer :: n=6
    real(tipo), dimension(n/2) :: diag_pri
    real(tipo), dimension(n/2-1) :: diag_inf
    call cria_matriz(n, diag_pri, diag_inf)

SUBROUTINE calcula_raizes_slapack(n, diag_pri, &
diag_inf)
  Calcula as raízes positivas do polinômio de Legendre
  de grau n (n deve ser par) em precisão simples

  Sintaxe:
  call calcula_raizes_slapack(n, diag_pri, diag_inf)

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss_Legendre. A ordem deve ser um natural par.
```

```
  diag_inf: vetor real, n/2-1 elementos; contém os
  elementos da diagonal inferior

  Argumentos de entrada e saída:
  diag_pri: vetor real, n/2 elementos; contém os
  elementos da diagonal principal, e onde ficam
  armazenadas as raízes positivas do polinômio de
  Legendre de grau n

  Descrição:
  Calcula as raízes positivas do polinômio de Legendre
  de grau n através de um problema de autovalor em
  precisão simples.
  O problema de autovalor é resolvido através da rotina
  SSTEVD da biblioteca LAPACK.

  Exemplos:
    integer :: n=4
    real(tipo), dimension(n/2) :: diag_pri
    real(tipo) :: diag_inf
    diag_pri=(/1.0_tipo/3.0_tipo, 11.0_tipo/21.0_tipo)
    diag_inf = (/sqrt(12.0_tipo/175.0_tipo)
    call calcula_raizes_slapack(n, diag_pri, diag_inf)

SUBROUTINE calcula_raizes_dlapack(n, diag_pri, &
diag_inf)
  Calcula as raízes positivas do polinômio de Legendre
  de grau n (n deve ser par) em precisão dupla

  Sintaxe:
  call calcula_raizes_dlapack(n, diag_pri, diag_inf)

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss_Legendre. A ordem deve ser um natural par.
  diag_inf: vetor real, n/2-1 elementos; contém os
  elementos da diagonal inferior
```

```
  Argumentos de entrada e saída:
  diag_pri: vetor real, n/2 elementos; contém os
  elementos da diagonal principal, e onde ficam
  armazenadas as raízes positivas do polinômio de
  Legendre de grau n

  Descrição:
  Calcula as raízes positivas do polinômio de Legendre
  de grau n através de um problema de autovalor em
  precisão dupla.
  O problema de autovalor é resolvido através da rotina
  DSTEVD da biblioteca LAPACK.

  Exemplos:
    integer :: n=4
    real(tipo), dimension(n/2) :: diag_pri
    real(tipo) :: diag_inf
    diag_pri=(/1.0_tipo/3.0_tipo, 11.0_tipo/21.0_tipo)
    diag_inf = sqrt(12.0_tipo/175.0_tipo)
    call calcula_raizes_dlapack(n, diag_pri, diag_inf)

SUBROUTINE calcula_raizes(n, diag_pri, diag_inf)
  Calcula as raízes positivas do polinômio de Legendre
  de grau n (n deve ser par) em precisão simples,
  dupla ou quádrupla

  Sintaxe:
  call calcula_raizes(n, diag_pri, diag_inf)

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss_Legendre. A ordem deve ser um natural par.
  diag_inf: vetor real, n/2-1 elementos; contém os
  elementos da diagonal inferior

  Argumentos de entrada e saída:
  diag_pri: vetor real, n/2 elementos; contém os
  elementos da diagonal principal, e onde ficam
```

```
  armazenadas as raízes positivas do polinômio de
  Legendre de grau n

  Descrição:
  Calcula as raízes positivas do polinômio de Legendre
  de grau n através de um problema de autovalor em
  precisão simples, dupla ou quádrupla.
  O problema de autovalor é resolvido através da rotina
  iter_qr.

  Exemplos:
    integer :: n=4
    real(tipo), dimension(n/2) :: diag_pri
    real(tipo) :: diag_inf
    diag_pri=(/1.0_tipo/3.0_tipo, 11.0_tipo/21.0_tipo)
    diag_inf=sqrt(12.0_tipo/175.0_tipo)
    call calcula_raizes(n, diag_pri, diag_inf)

SUBROUTINE pesos_gl(n, nos, pesos)
  Calcula os pesos da quadratura de Gauss-Legendre

  Sintaxe:
  call pesos_gl(n, nos, pesos)

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura gaussiana
  (n deve ser par)
  nos: vetor real, n/2 elementos; contém as raízes
  positivas do polinômio de Legendre de grau n

  Argumentos de saída:
  pesos: vetor real, n/2 elementos; contém os pesos
  da quadratura gaussiana

  Descrição:
  Calcula os pesos da quadratura de Gauss-Legendre
  através de um somatório.
```

```
  Exemplos:
    integer :: n = 4
    real(tipo), dimension(n/2) :: nos, pesos
    nos(1)=0.339981_tipo; nos(2)=0.861136_tipo
    call pesos_gl(n, nos, pesos)

SUBROUTINE gauss_legendre_slapack(n, nos, w)
  Calcula os nós positivos e respectivos pesos da
  quadratura de Gauss-Legendre de ordem n
  (n deve ser par) em precisão simples

  Sintaxe:
  call gauss_legendre_slapack(n, nos, w)

  Utiliza as seguintes subrotinas auxiliares:
  cria_matriz
  calcula_raizes_slapack
  pesos_gl

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura gaussiana
  (n deve ser par)

  Argumentos de saída:
  nos: vetor real, n/2 elementos; contém as raízes
  positivas do polinômio de Legendre de grau n
  pesos: vetor real, n/2 elementos; contém os pesos
  da quadratura gaussiana

  Descrição:
  Calcula os nós e pesos da quadratura de Gauss-Legendre
  de ordem n (n deve ser par) em precisão simples.
  O problema de autovalores para obtenção dos nós é
  resolvido pela rotina SSTEVD da biblioteca LAPACK, e
  os pesos são calculados através de um somatório.

  Exemplos:
    integer :: n = 6
```

```
    real(tipo), dimension(n/2) :: nos, pesos
    call gauss_legendre_slapack(n, nos, w)

SUBROUTINE gauss_legendre_dlapack(n, nos, w)
  Calcula os nós positivos e respectivos pesos da
  quadratura de Gauss-Legendre de ordem n (n deve
  ser par) em precisão dupla

  Sintaxe:
  call gauss_legendre_dlapack(n, nos, w)

  Utiliza as seguintes subrotinas auxiliares:
  cria_matriz
  calcula_raizes_dlapack
  pesos_gl

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura gaussiana
  (n deve ser par)

  Argumentos de saída:
  nos: vetor real, n/2 elementos; contém as raízes
  positivas do polinômio de Legendre de grau n
  pesos: vetor real, n/2 elementos; contém os pesos da
  quadratura gaussiana

  Descrição:
  Calcula os nós e pesos da quadratura de Gauss-Legendre
  de ordem n (n deve ser par) em precisão dupla.
  O problema de autovalores para obtenção dos nós é
  resolvido pela rotina DSTEVD da biblioteca LAPACK, e
  os pesos são calculados através de um somatório.

  Exemplos:
    integer :: n = 6
    real(tipo), dimension(n/2) :: nos, pesos
    call gauss_legendre_dlapack(n, nos, w)
```

```
SUBROUTINE gauss_legendre(n, nos, w)
  Calcula os nós positivos e respectivos pesos da
  quadratura de Gauss-Legendre de ordem n (n deve ser
  par) em precisão simples, dupla ou quádrupla

  Sintaxe:
  call gauss_legendre(n, nos, w)

  Utiliza as seguintes subrotinas auxiliares:
  cria_matriz
  calcula_raizes
  pesos_gl

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura gaussiana
  (n deve ser par)

  Argumentos de saída:
  nos: vetor real, n/2 elementos; contém as raízes
  positivas do polinômio de Legendre de grau n
  pesos: vetor real, n/2 elementos; contém os pesos
  da quadratura gaussiana

  Descrição:
  Calcula os nós e pesos da quadratura de Gauss-Legendre
  de ordem n (n deve ser par) em precisão simples,
  dupla ou quádrupla.
  O problema de autovalores para obtenção dos nós é
  resolvido pela rotina iter_qr, e os pesos são
  calculados através de um somatório.

  Exemplos:
    integer :: n = 6
    real(tipo), dimension(n/2) :: nos, pesos
    call gauss_legendre(n, nos, w)

SUBROUTINE quad_pntn_slapack(mu, eta, xi, pesos, n)
  Gera as direções de quadratura e os respectivos
```

```
pesos da quadratura PnTn para o hemisfério superior
da esfera unitária em precisão simples

Sintaxe:
call quad_pntn_slapack(vet_mu, vet_eta, vet_xi, &
vet_pesos, n)

Utiliza a seguinte função auxiliar:
gauss_legendre_slapack

Argumentos de entrada:
n: valor inteiro; ordem da quadratura de
Gauss-Legendre (a ordem deve ser par).

Argumentos de saída:
vet_mu: vetor real, n*n elementos; armazena a
direção mu
vet_eta: vetor real, n*n elementos; armazena a
direção eta
vet_xi: vetor real, n*n elementos; armazena a
direção xi
vet_pesos: vetor real, n*n elementos; armazena os
pesos

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
PnTn em precisão simples para o hemisfério superior
da esfera unitária.
Cada vetor tem tamanho n*n, sendo que o número de
direções discretas por octante da esfera unitária é
M = n*n/4.
A soma dos pesos de quadratura é 4*pi.
O problema de autovalores para a quadratura de
Gauss-Legendre é resolvido através da rotina SSTEVD
da biblioteca LAPACK, e os respectivos pesos são
calculados através de um somatório.

Exemplos:
```

```
    integer :: n = 6
    real(tipo), dimension(n*n) :: mu, eta, xi, pesos
    call quad_pntn_slapack(mu, eta, xi, pesos, n)

SUBROUTINE quad_pntn_dlapack(mu, eta, xi, pesos, n)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura PnTn para o hemisfério superior
  da esfera unitária em precisão dupla

  Sintaxe:
  call quad_pntn_dlapack(vet_mu, vet_eta, vet_xi, &
  vet_pesos, n)

  Utiliza a seguinte função auxiliar:
  gauss_legendre_dlapack

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss-Legendre (a ordem deve ser par).

  Argumentos de saída:
  vet_mu: vetor real, n*n elementos; armazena a
  direção mu
  vet_eta: vetor real, n*n elementos; armazena a
  direção eta
  vet_xi: vetor real, n*n elementos; armazena a
  direção xi
  vet_pesos: vetor real, n*n elementos; armazena os
  pesos

  Descrição:
  Gera as vetores mu, eta, xi e pesos da quadratura
  PnTn em precisão dupla para o hemisfério superior da
  esfera unitária.
  Cada vetor tem tamanho n*n, sendo que o número de
  direções por octante da esfera unitária é M = n*n/4.
  A soma dos pesos de quadratura é 4*pi.
  O problema de autovalores para a quadratura de
```

```
  Gauss-Legendre é resolvido através da rotina DSTEVD
  da biblioteca LAPACK, e os respectivos pesos são
  calculados através de um somatório.

  Exemplos:
    integer :: n = 6
    real(tipo), dimension(n*n) :: mu, eta, xi, pesos
    call quad_pntn_dlapack(mu, eta, xi, pesos, n)

SUBROUTINE quad_pntn(mu, eta, xi, pesos, n)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura PnTn para o hemisfério superior
  da esfera unitária em precisão simples, dupla ou
  quádrupla

  Sintaxe:
  call quad_pntn(vet_mu, vet_eta, vet_xi, vet_pesos, n)

  Utiliza a seguinte função auxiliar:
  gauss_legendre

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss-Legendre (a ordem deve ser par).

  Argumentos de saída:
  vet_mu: vetor real, n*n elementos; armazena a
  direção mu
  vet_eta: vetor real, n*n elementos; armazena a
  direção eta
  vet_xi: vetor real, n*n elementos; armazena a
  direção xi
  vet_pesos: vetor real, n*n elementos; armazena os
  pesos

  Descrição:
  Gera as vetores mu, eta, xi e pesos da quadratura
  PnTn em precisão simples, dupla ou quádrupla para o
```

```
  hemisfério superior da esfera unitária.
  Cada vetor tem tamanho n*n, sendo que o número de
  direções discretas por octante da esfera unitária é
  M = n*n/4.
  A soma dos pesos de quadratura é 4*pi.
  O problema de autovalores para a quadratura de
  Gauss-Legendre é resolvido através da rotina iter_qr,
  e os respectivos pesos são calculados através de um
  somatório.

  Exemplos:
    integer :: n = 6
    real(tipo), dimension(n*n) :: mu, eta, xi, pesos
    call quad_pntn(mu, eta, xi, pesos, n)

SUBROUTINE quad_pntnsn_slapack(mu, eta, xi, pesos, n)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura PnTnSn para o hemisfério superior
  da esfera unitária em precisão simples

  Sintaxe:
  call quad_pntnsn_slapack(vet_mu, vet_eta, vet_xi, &
  vet_pesos, n)

  Utiliza a seguinte função auxiliar:
  gauss_legendre_slapack

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss-Legendre (a ordem deve ser par).

  Argumentos de saída:
  vet_mu: vetor real, n*(n+2)/2 elementos; armazena
  a direção mu
  vet_eta: vetor real, n*(n+2)/2 elementos; armazena
  a direção eta
  vet_xi: vetor real, n*(n+2)/2 elementos; armazena
  a direção xi
```

```
  vet_pesos: vetor real, n*(n+2)/2 elementos; armazena
  os pesos

  Descrição:
  Gera as vetores mu, eta, xi e pesos da quadratura
  PnTnSn em precisão simples para o hemisfério
  superior da esfera unitária.
  Cada vetor tem tamanho n*(n+2)/2, sendo que o número
  de direções discretas por octante  da esfera unitária
  é M = n*(n+2)/8.
  A soma dos pesos de quadratura é 4*pi.
  O problema de autovalores para a quadratura de
  Gauss-Legendre é resolvido através da rotina SSTEVD
  da biblioteca LAPACK, e os respectivos pesos são
  calculados através de um somatório.

  Exemplos:
    integer :: n = 6
    real(tipo), dimension(n*(n+2)/2) :: mu, eta, xi, &
    pesos
    call quad_pntnsn_slapack(mu, eta, xi, pesos, n)

SUBROUTINE quad_pntnsn_dlapack(mu, eta, xi, pesos, n)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura PnTnSn para o hemisfério
  superior da esfera unitária em precisão dupla

  Sintaxe:
  call quad_pntnsn_dlapack(vet_mu, vet_eta, vet_xi, &
  vet_pesos, n)

  Utiliza a seguinte função auxiliar:
  gauss_legendre_dlapack

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss-Legendre (a ordem deve ser par).
```

```
  Argumentos de saída:
  vet_mu: vetor real, n*(n+2)/2 elementos; armazena
  a direção mu
  vet_eta: vetor real, n*(n+2)/2 elementos; armazena
  a direção eta
  vet_xi: vetor real, n*(n+2)/2 elementos; armazena
  a direção xi
  vet_pesos: vetor real, n*(n+2)/2 elementos; armazena
  os pesos

  Descrição:
  Gera as vetores mu, eta, xi e pesos da quadratura
  PnTnSn em precisão dupla para o hemisfério superior
  da esfera unitária.
  Cada vetor tem tamanho n*(n+2)/2, sendo que o número
  de direções discretas por octante da esfera unitária
  é M = n*(n+2)/8.
  A soma dos pesos de quadratura é 4*pi.
  O problema de autovalores para a quadratura de
  Gauss-Legendre é resolvido através da rotina DSTEVD
  da biblioteca LAPACK, e os respectivos pesos são
  calculados através de um somatório.

  Exemplos:
    integer :: n = 6
    real(tipo), dimension(n*(n+2)/2) :: mu, eta, xi, &
    pesos
    call quad_pntnsn_dlapack(mu, eta, xi, pesos, n)

SUBROUTINE quad_pntnsn(mu, eta, xi, pesos, n)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura PnTnSn para o hemisfério superior
  da esfera unitária em precisão simples, dupla ou
  quádrupla

  Sintaxe:
  call quad_pntnsn(vet_mu, vet_eta, vet_xi, &
  vet_pesos, n)
```

```
  Utiliza a seguinte função auxiliar:
  gauss_legendre

  Argumentos de entrada:
  n: valor inteiro; ordem da quadratura de
  Gauss-Legendre (a ordem deve ser par).

  Argumentos de saída:
  vet_mu: vetor real, n*(n+2)/2 elementos; armazena
  a direção mu
  vet_eta: vetor real, n*(n+2)/2 elementos; armazena
  a direção eta
  vet_xi: vetor real, n*(n+2)/2 elementos; armazena
  a direção xi
  vet_pesos: vetor real, n*(n+2)/2 elementos; armazena
  os pesos

  Descrição:
  Gera as vetores mu, eta, xi e pesos da quadratura
  PnTnSn em precisão simples, dupla ou quádrupla para
  o hemisfério superior da esfera unitária.
  Cada vetor tem tamanho n*(n+2)/2, sendo que o número
  de direções discretas por octante da esfera unitária
  é M = n*(n+2)/8.
  A soma dos pesos de quadratura é 4*pi.
  O problema de autovalores para a quadratura de
  Gauss-Legendre é resolvido através da rotina iter_qr,
  e os respectivos pesos são calculados através de um
  somatório.

  Exemplos:
    integer :: n = 6
    real(tipo), dimension(n*(n+2)/2) :: mu, eta, xi, &
    pesos
    call quad_pntnsn(mu, eta, xi, pesos, n)

SUBROUTINE iter_qr(n, diag_pri, diag_inf)
```

```
Calcula os autovalores de uma matriz tridiagonal
simétrica através do método QR implícito shift de
Wilkinson

Sintaxe:
call iter_qr(n, diag_pri, diag_inf)

Utiliza as seguintes funções auxiliares:
qr_imp
ordena

Argumentos de entrada:
n: valor inteiro; ordem da matriz simétrica
tridiagonal

Argumentos de entrada e saída:
diag_pri: vetor real, n elementos; contém a
diagonal principal da matriz tridiagonal simétrica
diag_inf: vetor real, n-1 elementos; contém a
subdiagonal da matriz tridiagonal simétrica

Descrição:
Este algoritmo é uma implementação do algoritmo
8.3.3 de Golub e Van Loan (2013, p. 463), para o
cálculo de autovalores de uma matriz tridiagonal
simétrica através do método QR implícito shift
de Wilkinson.
Os elementos da diagonal da matriz são armazenados
no vetor diag_pri e os elementos da subdiagonal da
matriz são armazenados no vetor diag_inf.
No final do processo, os autovalores estão
armazenados no vetor diag_pri.

Exemplos:
  integer :: n = 7
  real(tipo), dimension(n) :: diag_pri
  real(tipo), dimension(n-1) :: diag_inf
  diag_pri = 2.0_tipo
```

```
    diag_inf = 1.0_tipo
    call iter_qr(n, diag_pri, diag_inf)

SUBROUTINE qr_imp(n, diag_pri, diag_inf)
  Calcula uma iteração do método QR implícito com
  shift de Wilkilson

  Sintaxe:
  call qr_imp(n, diag_pri, diag_inf)

  Argumentos de entrada:
  n: valor inteiro; ordem da matriz simétrica
  tridiagonal

  Argumentos de entrada e saída:
  diag_pri: vetor real, n elementos; contém a
  diagonal principal da matriz tridiagonal simétrica
  diag_inf: vetor real, n-1 elementos; contém a
  subdiagonal da matriz tridiagonal simétrica

  Descrição:
  Este algoritmo é uma implementação do algoritmo
  8.3.2 de Golub e Van Loan (2013, p. 462), para uma
  iteração do método QR implícito com shift de
  Wilkinson.
  Esta iteração é calculada utilizando rotações de
  Givens, de acordo com Golub e Van Loan (2013).

  Exemplos:
    integer :: n = 7
    real(tipo), dimension(n) :: diag_pri
    real(tipo), dimension(n-1) :: diag_inf
    diag_pri = 2.0_tipo
    diag_inf = 1.0_tipo
    call qr_imp(n, diag_pri, diag_inf)

SUBROUTINE sinal(r, aux)
  Determina o sinal de r, se é positivo ou negativo
```

```
  Sintaxe:
  call sinal(r, aux)

  Argumentos de entrada:
  r: valor real; número do qual queremos descobrir
  o sinal

  Argumentos de saída:
  aux: valor inteiro; sinal do número r.
    aux = 1: o número r é positivo;
    aux = -1: o número r é negativo.

  Descrição:
  Determina o sinal do número real r. Caso o número r
  seja positivo, retorna aux=1; caso contrário,
  retorna aux=-1.

  Exemplos:
    integer::aux
    real(tipo):: r=2.0_tipo
    call sinal(r, aux)

SUBROUTINE ordena(x, n)
  Ordena o vetor x, do menor elemento ao maior elemento

  Sintaxe:
  call ordena (x, n)

  Utiliza as seguintes funções auxiliares:
  troca

  Argumentos de entrada:
  n: valor inteiro; tamanho do vetor x

  Argumentos de entrada e saída:
  x: vetor real, n elementos; vetor a ser ordenado
```

```
  Descrição:
  Ordena o vetor x, do menor ao maior elemento,
  através do método Bubblesort.

  Exemplos:
    integer :: n = 7
    real(tipo), dimension(n) :: x
    x = rand(7)
    call ordena(x, n)

SUBROUTINE troca(x, y)
  Troca os elementos x e y

  Sintaxe:
  call troca(x, y)

  Argumentos de entrada e saída:
  x, y: valores reais a serem trocados

  Descrição:
  Troca os valores dos números x, y. Ao final do
  processo, o valor inicial de x está armazenado em y
  e vice-versa.

  Exemplos:
    real(tipo) :: x=4.0_tipo, y=2.0_tipo
    call troca(x, y)

SUBROUTINE ordena_x(mu, eta, xi, pesos, M)
  Ordenamento ADO dos vetores mu, eta, xi e pesos
  para o esquema x

  Sintaxe:
  call ordena_x(mu, eta, xi, pesos, M)

  Utiliza as seguintes funções auxiliares:
  troca
```

```
  Argumentos de entrada:
  M: valor inteiro; número de direções discretas por
  octante

  Argumentos de entrada e saída:
  mu: vetor real, 4*M elementos; contém a direção mu
  eta: vetor real, 4*M elementos; contém a direção eta
  xi: vetor real, 4*M elementos; contém a direção xi
  pesos: vetor real, 4*M elementos; contém os pesos
  de quadratura

  Descrição:
  Rotina baseada no Bubblesort para ordenarmento ADO
  dos vetores mu, eta, xi e pesos para o esquema x.
  O ordamento é realizado pelo vetor xi.
  Para o funcionamento da rotina, é necessário que as
  primeiras M componentes de cada vetor sejam as
  componentes do octante principal.

  Exemplos:
   integer :: M = 12
    real(tipo), dimension(4*M) :: mu, eta, xi, pesos
    mu = rand(4*M)
    eta = rand(4*M)
    xi = rand(4*M)
    pesos= rand(4*M)
    call ordena_x(mu, eta, xi, pesos, M)

SUBROUTINE ordena_y(mu, eta, xi, pesos, M)
  Ordenamento ADO dos vetores mu, eta, xi e pesos para
  o esquema y

  Sintaxe:
  call ordena_y(mu, eta, xi, pesos, M)

  Utiliza as seguintes funções auxiliares:
  troca
```

```
  Argumentos de entrada:
  M: valor inteiro; número de direções discretas por
  octante

  Argumentos de entrada e saída:
  mu: vetor real, 4*M elementos; contém a direção mu
  eta: vetor real, 4*M elementos; contém a direção eta
  xi: vetor real, 4*M elementos; contém a direção xi
  pesos: vetor real, 4*M elementos; contém os pesos
  de quadratura

  Descrição:
  Rotina baseada no Bubblesort para ordenarmento ADO
  dos vetores mu, eta, xi e pesos para o esquema y.
  O ordamento é realizado pelo vetor xi.
  Para o funcionamento da rotina, é necessário que as
  primeiras M componentes de cada vetor sejam as
  componentes do octante principal.

  Exemplos:
   integer :: M = 12
    real(tipo), dimension(4*M) :: mu, eta, xi, pesos
    mu = rand(4*M)
    eta = rand(4*M)
    xi = rand(4*M)
    pesos= rand(4*M)
    call ordena_y(mu, eta, xi, pesos, M)

SUBROUTINE fn_qrs_full_tanh_sinh_sing(xi, wi, n1, n2, &
phi0, h, tol)
  Calcula os nós e pesos da quadratura QR (para a
  variável azimutal) via polinômios ortogonais em
  precisão simples, dupla ou quádrupla, no caso em que
  temos singularidade azimutal em phi = phi0

  Sintaxe:
  call fn_qrs_full_tanh_sinh_sing(xi, wi, n1, n2, &
  phi0, h, tol)
```

```
Utiliza as seguintes funções auxiliares:
fn_tanh_sinh_quad
fn_orpolyval
iter_qr
calcula_pesos

Argumentos de entrada:
n1: valor inteiro; ordem da quadratura para o
intervalo [0, phi0)
n2: valor inteiro; ordem da quadratura para o
intervalo (phi0, pi/2]
phi0: valor real; ponto de singularidade azimutal
h: valor real; passo da quadratura tanh-sinh
tol: valor real; tolerância quadratura tanh-sinh

Argumentos de saída:
xi: vetor real, n=n1+n2 elementos; contém os nós
da quadratura
wi: vetor real, n=n1+n2 elementos; contém os pesos
da quadratura

Descrição:
Calcula os nós e pesos da quadratura QR (para a
variável azimutal) via polinômios ortogonais em
precisão simples, dupla ou quádrupla, no caso em que
temos singularidade azimutal em phi = phi0.
O vetor nós e pesos possuem o total de n=n1+n2
elementos, de modo que, xi(1:n1) e wi(1:n1) armazenam
os nós e pesos associados ao intervalo [0, phi0);
xi(n1+1:n1+n2) e xi(n1+1:n1+n2) armazenam os nós e
pesos associado ao intervalo (phi0, pi/2].
Os nós são obtidos resolvendo um problema de
autovalores para uma matriz tridiagonal simétrica,
através da rotina iter_qr, que foi implementada por
nós. Os pesos são calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
```

```
  quadratura tanh-sinh.

  Exemplos:
    integer :: n1=2, n2=3
    real(tipo), dimension(n1+n2) :: xi, wi
    real(tipo) :: h=0.004_tipo, tol=1e-14, phi0
    phi0 = acos(-1.0_tipo)/3.0_tipo
    call fn_qrs_full_tanh_sinh_sing(xi, wi, n1, n2, &
    phi0, h, tol)

SUBROUTINE quad_qr_tanh_sinh_sing(mu, eta, xi, &
pesos, ordem_polar, ordem_azi1, ordem_azi2, phi0, &
h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR hemisfério superior da esfera
  unitária, no caso em que temos singularidade azimutal
  em phi=phi0, em precisão simples, dupla ou quádrupla

  Sintexe:
  call quad_qr_tanh_sinh_sing(mu, eta, xi, pesos, &
  ordem_polar, ordem_azi1, ordem_azi2, phi0, h, tol)

  Utiliza as seguintes funções auxiliares:
  fn_qrs_full_tanh_sinh
  fn_qrs_full_tanh_sinh_sing

  Argumentos de entrada:
  ordem_polar: valor inteiro; determina a ordem da
  quadratura polar
  ordem_azi1: valor inteiro; determina a ordem da
  quadratura azimutal para o intervalo [0, phi0)
  ordem_azi2: valor inteiro; determina a ordem da
  quadratura azimutal para o intervalo (phi0, pi/2]
  phi0: valor real; ponto de singularidade azimutal
  h: valor real; passo da quadratura tanh-sinh
  tol: valor real; tolerância quadratura tanh-sinh

  Argumentos de saída:
```

```
mu: vetor real, 4*ordem_polar*(ordem_azi1+ordem_azi2)
elementos; armazena a direção mu da quadratura
eta: vetor real, 4*ordem_polar*(ordem_azi1+ordem_azi2)
elementos; armazena a direção eta da quadratura
xi: vetor real, 4*ordem_polar*(ordem_azi1+ordem_azi2)
elementos; armazena a direção xi da quadratura
pesos: vetor real, 4*ordem_polar*(ordem_azi1+
ordem_azi2) elementos; armazena os pesos da quadratura

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
QR com singularidade azimutal em phi=phi0, em
precisão simples, dupla ou quádrupla,  para o
hemisfério superior da esfera unitária.
Cada vetor tem tamanho 4*ordem_polar*(ordem_azi1+
ordem_azi2), /sendo que o número de direções
discretas por octante da esfera unitária é
M = ordem_polar*(ordem_azi1+ordem_azi2).
É utilizada a mesma ordem azimutal em todos os níveis
polares.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
O problema de autovalores para as quadraturas
gaussianas é resolvido pela da rotina iter_qr, que
foi implementada por nós.
Os pesos são calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura tanh-sinh.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol, phi0
  integer :: ordem_polar, ordem_azi1, ordem_azi2, M
  h = 0.004_tipo; tol = 1e-16
  phi0 = acos(-1.0_tipo)/3.0_tipo
```

```
    ordem_polar = 5; ordem_azi1 = 3; ordem_azi2 = 4
    M = ordem_polar*(ordem_azi1+ordem_azi2)
    allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
    call quad_qr_tanh_sinh_sing(mu, eta, xi, pesos, &
    ordem_polar, ordem_azi1, ordem_azi2, phi0, h, tol)

SUBROUTINE fn_qrs_full_erf_sing(xi, wi, n1, n2, phi0, &
h, tol)
  Calcula os nós e pesos da quadratura QR (para a
  variável azimutal) via polinômios ortogonais em
  precisão simples, dupla ou quádrupla, no caso em que
  temos singularidade azimutal em phi = phi0

  Sintaxe:
  call fn_qrs_full_erf_sing(xi, wi, n1, n2, phi0, &
  h, tol)

  Utiliza as seguintes funções auxiliares:
  fn_erfun_quad
  fn_orpolyval
  iter_qr
  calcula_pesos

  Argumentos de entrada:
  n1: valor inteiro; ordem da quadratura para o
  intervalo [0, phi0)
  n2: valor inteiro; ordem da quadratura para o
  intervalo (phi0, pi/2]
  phi0: valor real; ponto de singularidade azimutal
  h: valor real; passo da quadratura função erro
  tol: valor real; tolerância quadratura função erro

  Argumentos de saída:
  xi: vetor real, n=n1+n2 elementos; contém os nós
  da quadratura
  wi: vetor real, n=n1+n2 elementos; contém os pesos
  da quadratura
```

```
  Descrição:
  Calcula os nós e pesos da quadratura QR (para a
  variável azimutal) via polinômios ortogonais em
  precisão simples, dupla ou quádrupla, no caso em que
  temos singularidade azimutal em phi = phi0.
  O vetor nós e pesos possuem o total de n=n1+n2
  elementos, de modo que, xi(1:n1) e wi(1:n1)
  armazenam os nós e pesos associados ao intervalo
  [0, phi0); xi(n1+1:n1+n2) e xi(n1+1:n1+n2) armazenam
  os nós e pesos associado ao intervalo (phi0, pi/2].
  Os nós são obtidos resolvendo um problema de
  autovalores para uma matriz tridiagonal simétrica,
  através da rotina iter_qr, que foi implementada por
  nós. Os pesos são calculados através de um somatório.
  Os coeficientes da relação de recorrência que geram
  os polinômios ortogonais são calculados através da
  quadratura função erro.

  Exemplos:
    integer :: n1=2, n2=3
    real(tipo), dimension(n1+n2) :: xi, wi
    real(tipo) :: h=0.004_tipo, tol=1e-14, phi0
    phi0 = acos(-1.0_tipo)/3.0_tipo
    call fn_qrs_full_erf_sing(xi, wi, n1, n2, phi0, &
    h, tol)

SUBROUTINE quad_qr_erf_sing(mu, eta, xi, pesos, &
ordem_polar, ordem_azi1, ordem_azi2, phi0, h, tol)
  Gera as direções de quadratura e os respectivos
  pesos da quadratura QR hemisfério superior da esfera
  unitária, no caso em que temos singularidade azimutal
  em phi=phi0, em precisão simples, dupla ou quádrupla

  Sintexe:
  call quad_qr_erf_sing(mu, eta, xi, pesos, &
  ordem_polar, ordem_azi1, ordem_azi2, phi0, h, tol)

  Utiliza as seguintes funções auxiliares:
```

```
fn_qrs_full_erf
fn_qrs_full_erf_sing

Argumentos de entrada:
ordem_polar: valor inteiro; determina a ordem da
quadratura polar
ordem_azi1: valor inteiro; determina a ordem da
quadratura azimutal para o intervalo [0, phi0)
ordem_azi2: valor inteiro; determina a ordem da
quadratura azimutal para o intervalo (phi0, pi/2]
phi0: valor real; ponto de singularidade azimutal
h: valor real; passo da quadratura função erro
tol: valor real; tolerância quadratura função erro

Argumentos de saída:
mu: vetor real, 4*ordem_polar*(ordem_azi1+ordem_azi2)
elementos; armazena a direção mu da quadratura
eta: vetor real, 4*ordem_polar*(ordem_azi1+ordem_azi2)
elementos; armazena a direção eta da quadratura
xi: vetor real, 4*ordem_polar*(ordem_azi1+ordem_azi2)
elementos; armazena a direção xi da quadratura
pesos: vetor real, 4*ordem_polar*(ordem_azi1+ordem_azi2)
elementos; armazena os pesos da quadratura

Descrição:
Gera as vetores mu, eta, xi e pesos da quadratura
QR com singularidade azimutal em phi=phi0, em
precisão simples, dupla ou quádrupla, para o
hemisfério superior da esfera unitária.
Cada vetor tem tamanho 4*ordem_polar*(ordem_azi1+
ordem_azi2), sendo que o número de direções discretas
por octante da esfera unitária é M = ordem_polar*
(ordem_azi1+ordem_azi2).
É utilizada a mesma ordem azimutal em todos os
níveis polares.
Temos theta \in [0, pi] e phi \in [0, 2pi], em que
theta é o ângulo polar medido a partir do eixo xi e
phi o ângulo azimutal medido a partir do eixo x.
```

```
O problema de autovalores para as quadraturas
gaussianas é resolvido pela da rotina iter_qr, que
foi implementada por nós.
Os pesos são calculados através de um somatório.
Os coeficientes da relação de recorrência que geram
os polinômios ortogonais são calculados através da
quadratura função erro.

Exemplos:
  real(tipo), dimension(:), allocatable :: mu, eta, &
  xi, pesos
  real(tipo):: h, tol, phi0
  integer :: ordem_polar, ordem_azi1, ordem_azi2, M
  h = 0.004_tipo; tol = 1e-16
  phi0 = acos(-1.0_tipo)/3.0_tipo
  ordem_polar = 5; ordem_azi1 = 3; ordem_azi2 = 4
  M = ordem_polar*(ordem_azi1+ordem_azi2)
  allocate(mu(4*M), eta(4*M), xi(4*M), pesos(4*M))
  call quad_qr_erf_sing(mu, eta, xi, pesos, &
  ordem_polar, ordem_azi1, ordem_azi2, phi0, h, tol)
```

APÊNDICE B: Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura no octante principal da esfera unitária

Tabela B.1 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $P_N T_N$

N	Erro relativo mínimo	Erro relativo máximo	N	Erro relativo mínimo	Erro relativo máximo
2	0	1	16	0	0.163086
4	0	1	20	0	0.0593356
8	0	0.955237	32	0	0.0214108
12	0	0.522712	64	0	0.00517555

Tabela B.2 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $P_N T_N S_N$

N	**Erro relativo mínimo**	**Erro relativo máximo**	**N**	**Erro relativo mínimo**	**Erro relativo máximo**
2	0	1	16	4.24074e-16	0.152666
4	1.41358e-16	1	20	1.06018e-15	0.0637908
8	0	0.955236	32	0	0.0236419
12	2.12037e-16	0.521067	64	0	0.00586614

Tabela B.3 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRS45m\_Q_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	4.24074e-16	1	8	4.7377e-16	0.545092
2	2.82716e-16	1	10	1.61148e-14	0.253823
4	1.76697e-16	0.998318	16	9.27662e-16	0.00936683
6	0	0.881521	32	2.10873e-15	4.16882e-10

Tabela B.4 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRA45m\_Q_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	1.41358e-16	1	8	0	0.622408
2	1.41358e-16	1	10	1.06018e-14	0.326914
4	1.41358e-16	0.99914	16	1.36057e-14	0.0199369
6	9.89506e-16	0.915831	32	3.98862e-16	2.14511e-08

Tabela B.5 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRJ45m\_Q_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	7.0679e-16	1	8	0	0.242669
2	0	1	10	1.48426e-14	0.0590398
4	0	0.981933	16	2.26349e-14	0.0214108
6	1.13086e-15	0.656802	32	8.35526e-16	0.00517555

Tabela B.6 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRS90m\_Q_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	1.41358e-16	1	8	1.23688e-16	0.454986
2	4.24074e-16	1	10	2.24759e-14	0.135735
4	0	0.976297	16	6.16014e-15	0.0439228
6	4.30805e-16	0.770536	32	0	0.0103489

Tabela B.7 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRJ90m\_Q_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	7.0679e-16	1	8	1.32523e-16	0.539806
2	1.13086e-15	1	10	1.68216e-14	0.254522
4	8.48148e-16	0.998045	16	3.53395e-16	0.0178078
6	4.24074e-16	0.87538	32	0	1.85504e-08

Tabela B.8 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRS45m\_T_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	4.24074e-16	1	8	2.82716e-16	0.543822
2	4.24074e-16	1	10	1.49839e-14	0.247024
4	4.24074e-16	0.998318	16	3.49861e-15	0.00562928
6	7.0679e-16	0.881507	32	1.81115e-15	2.41718e-08

Tabela B.9 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRA45m\_T_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	1.41358e-16	1	8	6.36111e-16	0.620927
2	0	1	10	1.35704e-14	0.318222
4	2.12037e-16	0.99914	16	1.70933e-14	0.0124296
6	1.27222e-15	0.915816	32	0	9.27159e-09

Tabela B.10 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRJ45m\_T_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	7.0679e-16	1	8	0	0.242202
2	0	1	10	1.48426e-14	0.0606038
4	0	0.981933	16	8.74653e-16	0.0225671
6	1.27222e-15	0.656794	32	3.27862e-16	0.00559483

Tabela B.11 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRS90m\_T_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	1.41358e-16	1	8	1.41358e-15	0.454106
2	7.0679e-16	1	10	1.71043e-14	0.140248
4	5.65432e-16	0.976297	16	1.7228e-16	0.0464196
6	1.41358e-15	0.770499	32	0	0.0111931

Tabela B.12 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, no octante principal, para o esquema $QRJ90m\_T_{N_\theta}$

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	7.0679e-16	1	8	5.65432e-16	0.537392
2	1.13086e-15	1	10	1.51253e-14	0.244159
4	2.82716e-16	0.998045	16	1.03368e-14	0.0112453
6	9.89506e-16	0.875342	32	0	1.16816e-08

APÊNDICE C: Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura na esfera unitária, nos casos em que l e m são pares

Tabela C.1 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $P_N T_N$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
2	0	1	16	0	0.163086
4	0	1	20	0	0.0360108
8	0	0.955237	32	0	3.964e-05
12	0	0.522712	64	0	1.11673e-14

Tabela C.2 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $P_N T_N S_N$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
2	0	1	16	0	0.152666
4	0	1	20	8.48148e-16	0.0266023
8	2.12037e-16	0.955236	32	8.48148e-16	3.33177e-06
12	0	0.521067	64	0	6.45501e-09

Tabela C.3 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRS45m\_Q_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	4.24074e-16	1	8	2.18663e-16	0.545092
2	2.82716e-16	1	10	1.75284e-14	0.253823
4	1.23688e-16	0.998318	16	2.31827e-14	0.00936683
6	2.47376e-16	0.881521	32	0	4.16883e-10

Tabela C.4 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRA45m\_Q_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	1.41358e-16	1	8	2.12037e-16	0.622408
2	1.41358e-16	1	10	1.21568e-14	0.326914
4	4.24074e-16	0.99914	16	1.54434e-14	0.0199369
6	8.48148e-16	0.915831	32	1.36095e-15	2.14511e-08

Tabela C.5 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRJ45m\_Q_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	7.0679e-16	1	8	0	0.242669
2	2.12037e-16	1	10	1.49839e-14	0.0568933
4	2.12037e-16	0.981933	16	2.51617e-14	6.90517e-05
6	9.89506e-16	0.656802	32	1.13086e-15	3.91575e-14

Tabela C.6 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRS90m\_Q_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	1.27222e-15	1	8	3.71065e-16	0.454986
2	6.36111e-16	1	10	2.14157e-14	0.125619
4	1.41358e-16	0.976297	16	1.53741e-14	0.00240572
6	6.55989e-16	0.770536	32	7.11813e-16	4.15362e-12

Tabela C.7 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRJ90m\_Q_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	7.0679e-16	1	8	3.97569e-16	0.535884
2	1.13086e-15	1	10	1.6963e-14	0.253689
4	8.48148e-16	0.998033	16	4.90474e-15	0.0177115
6	4.24074e-16	0.87538	32	2.24614e-16	1.85504e-08

Tabela C.8 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRS45m\_T_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	4.24074e-16	1	8	0	0.543822
2	2.82716e-16	1	10	1.55494e-14	0.247024
4	4.24074e-16	0.998318	16	2.31827e-14	0.00562928
6	9.89506e-16	0.881507	32	2.82588e-15	4.09204e-11

Tabela C.9 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRA45m\_T_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	1.41358e-16	1	8	8.48148e-16	0.620927
2	0	1	10	1.41358e-14	0.318222
4	4.24074e-16	0.99914	16	2.16278e-14	0.0124296
6	1.27222e-15	0.915816	32	2.87287e-16	2.82871e-09

Tabela C.10 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRJ45m\_T_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	7.0679e-16	1	8	2.12037e-16	0.242202
2	0	1	10	1.52667e-14	0.055586
4	0	0.981933	16	2.17691e-14	4.25978e-05
6	1.48426e-15	0.656794	32	8.48148e-16	4.00313e-14

Tabela C.11 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRS90m\_T_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	1.27222e-15	1	8	2.12037e-15	0.454106
2	1.83765e-15	1	10	1.66802e-14	0.122107
4	9.89506e-16	0.976297	16	1.60353e-14	0.00181817
6	1.6963e-15	0.770499	32	3.55907e-16	6.35019e-11

Tabela C.12 Valores mínimos e máximos do erro relativo entre o valor analítico e o da quadratura, na esfera unitária, para o esquema $QRJ90m\_T_{N_\theta}$, nos casos em que l e m são pares

N_θ	**Erro relativo mínimo**	**Erro relativo máximo**	N_θ	**Erro relativo mínimo**	**Erro relativo máximo**
1	7.0679e-16	1	8	8.48148e-16	0.533654
2	1.13086e-15	1	10	1.48426e-14	0.242892
4	5.65432e-16	0.998033	16	1.01071e-14	0.0109835
6	1.13086e-15	0.875342	32	1.62654e-16	2.54058e-09