Grades diádicas adaptativas para simulação de escoamento de petróleo

Claudio Guido S. Cardoso

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Grades diádicas adaptativas para simulação de escoamento de petróleo

Claudio Guido S. Cardoso

Outubro de 2004

Banca Examinadora:

- Prof. Dr. Jorge Stolfi
  IC - UNICAMP (Orientador)
- Prof. Dr. Denis J. Schiozer
  CEPETRO - UNICAMP
- Profa. Dra. Maria Cristina de Castro Cunha
  IMECC - UNICAMP
- Prof. Dr. Cid Carvalho de Souza
  IC - UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Cardoso, Claudio Guido Silva

C179g Grades diádicas adaptativas para simulação de escoamento de petróleo / Claudio Guido Silva Cardoso -- Campinas, [S.P. :s.n.], 2004.

Orientadores : Jorge Stolfi ; Anamaria Gomide

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Simulação (Computadores digitais). 2. Equações diferenciais. 3. Galerkin, Métodos de. 4. Reservatório de petróleo. I. Stolfi, Jorge. II. Gomide, Anamaria. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Grades diádicas adaptativas para simulação de escoamento de petróleo

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Claudio Guido S. Cardoso e aprovada pela Banca Examinadora.

Campinas, Novembro de 2004.

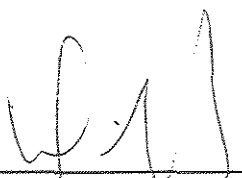
Prof. Dr. Jorge Stolfi
IC - UNICAMP (Orientador)

Profa. Dra. Anamaria Gomide
IC - UNICAMP (Co-orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 01 de dezembro de 2004, pela Banca examinadora composta pelos Professores Doutores:



Prof. Dr. Denis José Schiozer
CEPETRO - UNICAMP

Profa. Dra. Maria Cristina de Castro Cunha
IMECC - UNICAMP

Prof. Dr. Cid Carvalho de Souza
IC - UNICAMP

Prof. Dr. Jorge Stolfi
IC - UNICAMP

"A vida é um cubo. Não sei porque, mas é."

Conrado Gomes

Resumo

Nesta dissertação, estudamos as aplicações de *splines* construídos sobre grades diádicas na aproximação de funções e solução de equações diferenciais. Nosso foco principal é a simulação de escoamento de petróleo em reservatórios naturais.

Uma grade diádica é construída através de divisões sucessivas de uma célula raiz retangular, sempre pela metade, alternando a direção do corte ciclicamente pelos eixos de coordenadas. Desta forma, os *splines* diádicos podem ser refinados adaptativamente de acordo com as funções a serem aproximadas, até que satisfaçam o grau de precisão exigido. As grades diádicas também possuem uma estrutura muito simples e podem ser representadas e manipuladas computacionalmente com grande eficiência e facilidade.

Trabalhamos em particular com uma família de *splines* diádicos multilineares e contínuos para uma grade arbitrária G, que denominamos por $\mathcal{E}^1[G]$. Desenvolvemos algoritmos que geram duas bases padrões de elementos finitos (*tendas diádicas*) $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ para estes espaços. Para testarmos as potencialidades dos espaços $\mathcal{E}^1[G]$, realizamos experimentos numéricos utilizando a técnica de aproximação por mínimos quadrados, e resolvemos alguns casos de equações diferenciais parciais lineares. Nestes experimentos, estudamos o uso de malhas diádicas de resolução variável no espaço e no tempo. Concluímos que grades adaptativas podem produzir os mesmos resultados que uma grade uniforme fina, a uma fração do custo.

A última parte deste trabalho descreve em detalhes a construção e teste de um simulador para escoamento bifásico óleo/água (*Simóleo*), baseado em grades diádicas adaptativas. Revisamos, em linhas gerais, as equações que regem este processo, e sua discretização pelo método de elementos finitos (Galerkin). Desta forma, obtemos um sistema de equações diferenciais não linear, dependente do espaço e do tempo, que resolvemos de maneira iterativa. O desempenho do Simóleo foi avaliado utilizando um modelo popular de reservatório (5 poços).

Abstract

In this dissertation we study the application of splines built over dyadic grids for function approximation and integration of differential equations. Our main focus is the simulation of oil flow in natural reservoirs.

A dyadic grid is built by recursively splitting a rectangular root cell into equal halves, alternating the orientation of the cut cyclically over the coordinate axis. Therefore, dyadic splines may be refined adaptatively to fit the functions they are approximating, until a certain precision level is reached. Dyadic grids also have a simple structure which may be easily and efficiently represented and manipulated by computers.

We worked in particular with a family of multilinear and continuous dyadic splines for a certain grid G, denoted by $\mathcal{E}^1[G]$. Algorithms were developed to build two standard basis of finite elements (*dyadic tents*), $\mathcal{B}_{max}$ and $\mathcal{B}_{min}$, for such function spaces. In order to test the potentialities of the spaces $\mathcal{E}^1[G]$, we performed some numerical experiments of least squares approximations and integration of linear partial differential equations. In these experiments, we studied the use of adaptive dyadic grids, whose resolution varies in time and space. We conclude that adaptive grids can produce the same results as a fine uniform grid, at a fraction of the cost.

The last part of this work describes in detail the development and tests of an oil/water flow simulator (*Simóleo*), based on adaptive dyadic grids. We review the oil flow equations, and their discretization by the finite element (Galerkin) method. Thus, we obtain a non linear differential equation system, depending on space and time, which we solve iteratively. The program's performance is measured with experiments using a popular reservoir model (5-spot).

Agradecimentos

Para não fugir à regra, agradeço logo ao Divino por ter enfim chegado nesta parte da tese, e pela profunda paz de espírito que ele tem sempre dado a mim e aos meus. Obrigado também aos meus fantásticos pais Oscarito e Wilma, que me apóiam até mesmo quando resolvo deixá-los pra passar um tempo longe da velha Bahia. Um grande beijo pra minha querida irmã Carmélia (Mel), que perdeu um carnaval inteiro tomando chuva aqui comigo, em um dos períodos mais divertidos da minha estada em Campinas! Para comparar com estes dias, somente a visita do vô Cleber (Varvito) que fez com que me sentisse um pouco em casa no primeiro aniversário longe (e ainda trouxe um bolo da vó). Um enorme abraço pra vocês, vovôs Oscar e Cleber, vovós Lacy e Carmélia. Este livro é apenas um dos muitos frutos do que vocês semearam.

À minha princesa Priscila (a chú!), meus agradecimentos de coração pelo enorme carinho, paciência e companheirismo expressos no pouco tempo que passamos juntos, e ao longo de muitos e muitos dias do outro lado da linha.

Desejo tudo de bom para a dupla dinâmica de orientadores Stolfi e Ana, exímios computeiros, matemáticos, físicos e comedores de pizza! Sempre foi uma enorme satisfação trabalhar com vocês.

Um forte abraço aos irmãos pistolinhas: Zeh (pistolinha aposentado, churrasqueiro profissional e também o consumidor mais chat... digo, mais exigente do mundo), Triste Toy Boy (cuja gargalhada sutil tem até eco), Guto (miseravão e quem devo meu próprio título pistolístico), Daniel Linux (bora Baêêêaaaa!), Bart (pistolinha adjunto, o terrível), Glauber (pistolinha honorário, o bonecão de Olinda); e a todos os seus agradabilíssimos agregados: Carmen, Yeda, Celso, Mara, Marcão e o pessoal da física, meu recém-chegado sobrinho Gutinho, sua mamãe Tina, Mércia, Messias, Daniela e a todo o resto do povo!

Aos grandes amigos de caminhadas pelos corredores do IC: Marcão (Vinícius), Magrão, Bila, Bazinho, Pará, Silvana, Borin, Jú (do Borin), Jú (do Tio Sam), Márcio, Felipe (Ozzy), Silvania, Cesar (fala miseravation!), Jean, Amanda, Greg

Skywalker, Larissa, Lázaro, Chenca (e a todos os mardito), Daniel Andrade (e aí, novidades?!), Nielsen, Wesley, Daniela, Thaísa, Sandro, Luís, Nilton, Cleber, Evandro, Norton, Fernando (monstro), Fernando (do Pará), Luciana, Fábio, Schubert, Sheila, Marília, Erik, Luciano, os infalíveis vigias Nelson, Flávio, Karin, juntamente com os professores, pessoal da administração, e a todo mundo que ficou faltando na lista (agora é melhor correr atrás da próxima tese, meu chapa!), um axé caloroso, e nos vemos por aí!

Meus agradecimentos ao pessoal da engenharia de petróleo (Valmir, Eliana, prof. Denis), à profa. Cristina e a todos que ajudaram a fazer a água deste simulador andar!

E obrigado também à CAPES, CNPq e à UNICAMP pelo auxilio financeiro! (ufa! Essa foi por pouco!)

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Propomos neste trabalho o uso de bases de *splines* definidas sobre grades diádicas adaptativas para a aproximação de funções e solução de equações diferenciais, tendo como foco principal a simulação de escoamento de petróleo em reservatórios naturais.

1.1 *Splines* diádicos adaptativos

É comum usarmos espaços de aproximação para lidarmos com problemas envolvendo funções muito irregulares e equações diferenciais, porque desta forma conseguimos simplificá-los e manipulá-los por meio de computadores. Estes espaços podem conter por exemplo séries trigonométricas ou polinômios. Aqui utilizamos funções de aproximação muito populares: os *splines* [BBB87, BQDY89, Mor85], cujos domínios são divididos em regiões distintas, onde possuem comportamento essencialmente polinomial. Na figura 1.1 mostramos um *spline* unidimensional, linear em cada segmento do eixo x (esquerda), e um *spline* bidimensional em curvas de nível (direita).

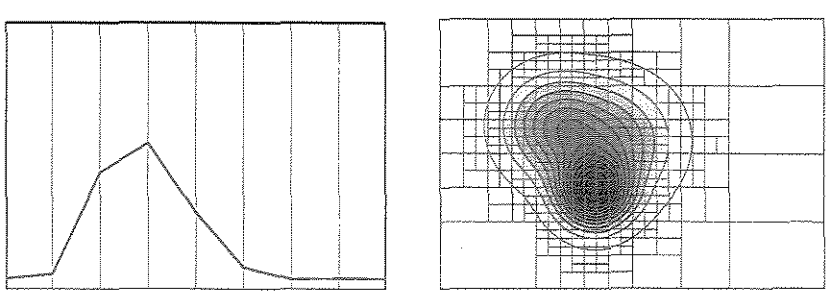


Figura 1.1: Exemplos de *splines* em uma e duas dimensões.

Trabalhamos especificamente com *splines diádicos*, ou *splines* definidos sobre *grades diádicas*. Uma grade diádica é construída basicamente através de divisões sucessivas de uma célula raiz retangular, sempre pela metade. A figura 1.2 abaixo mostra a grade diádica sobre a qual o *spline* bidimensional da figura 1.1 está definido.

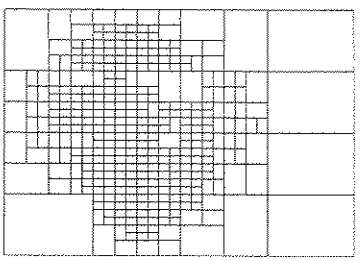


Figura 1.2: Exemplo de grade diádica bidimensional.

A vantagem do uso de grades diádicas é que elas possuem uma estrutura muito simples, e podem ser representadas e manipuladas computacionalmente com grande economia e facilidade. Ao mesmo tempo, podemos refinar um *spline* diádico até que ele esteja suficientemente próximo de uma função qualquer.

1.2 Simulação do escoamento de petróleo

Nossa principal motivação é investigar a aplicação de *splines* diádicos na simulação do escoamento de petróleo em reservatórios naturais [AS79, Cri77, Str88]. Esta simulação é complexa devido à presença de outros fluidos (água e/ou gás) que se movem com velocidades diferentes da do óleo. Além disso, métodos modernos de extração de petróleo utilizam o bombeamento de água ou outras substâncias em determinados poços de injeção, criando diferenças de pressão que empurram o óleo presente nos meios porosos subterrâneos na direção dos poços de extração (figura 1.3).

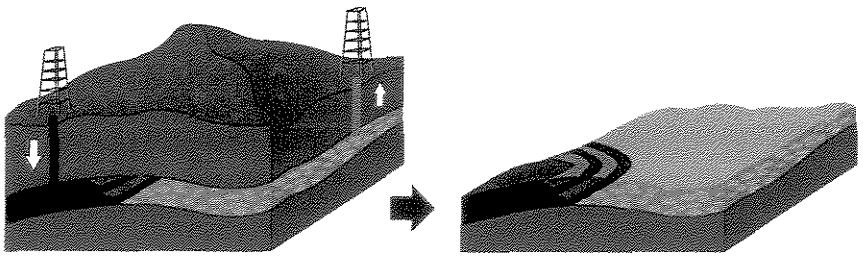


Figura 1.3: Processo de explotação de petróleo.

1.2.1 Diferenças finitas

O movimento dos fluidos num reservatório é descrito por equações diferenciais parciais *não lineares*. Os primeiros simuladores utilizaram o método de diferenças finitas, em que o estado do reservatório é amostrado numa grade regular. Este método continua sendo o mais usado nos simuladores comerciais.

1.2.2 Métodos de resolução variável

Porém, o escoamento geralmente inclui várias *regiões críticas*, onde as grandezas relevantes variam rapidamente no espaço. Essas regiões podem ser estáticas, como nas proximidades dos poços, falhas geológicas, aquíferos, etc; ou dinâmicas (variáveis no tempo), como as interfaces razoavelmente abruptas entre as substâncias móveis (óleo/água, óleo/gás, etc). A diferença de escala necessária para a simulação nestes pontos e no restante do domínio é muito grande e pode variar de dezenas de quilômetros a poucos metros.

Devido à extensão dos campos típicos, é inviável representar todo o reservatório na escala mais detalhada, ainda mais quando consideramos que é necessário executar centenas de simulações, abrangendo décadas de funcionamento.

Por esta razão, boa parte da pesquisa na área objetiva reduzir o custo computacional das simulações sem prejudicar significativamente a precisão dos resultados. As idéias propostas para este fim incluem o uso de sistemas híbridos de coordenadas [NLP+90], abordagens multiescalas [Gue98], simulação por linhas de fluxo [dCS98], entre outras. Neste sentido, uma das técnicas mais comuns é o *refinamento local*, na qual apenas as regiões críticas do reservatório são representadas com maior riqueza de detalhes [ATL91, GE97, HGH83, NLP+90, Ris02, Was87]. Muitas dessas abordagens sofrem de dois problemas: a natureza e representação da grade, que dificultam sua modificação dinâmica; e a restrição a poucos níveis

de refinamento, com escalas muito diferentes entre si, que implica em desperdício de trabalho.

1.2.3 Vantagens dos *splines* diádicos

Os *splines* diádicos proporcionam refinamento local dinâmico e gradual: o espaço de funções aproximadoras pode ser adaptado ao longo do processo de escoamento. Nos locais onde apresentam pouca variação, aproximamos as propriedades petrofísicas com segmentos mais grosseiros da grade diádica, acelerando os cálculos e economizando recursos do simulador.

1.3 Estrutura do trabalho

No capítulo 2 descrevemos em detalhes as grades diádicas, mostrando a correspondência entre elas e as árvores binárias. No capítulo 3 definimos os *splines* diádicos, associados a uma determinada grade diádica, e em particular os *splines* multilineares de continuidade zero, que utilizaremos nesta dissertação. Exibimos duas bases padrões de elementos finitos para este espaço, as bases $\mathcal{B}_{max}$ (maximal) e $\mathcal{B}_{min}$ (minimal), que nos permitem representar sucintamente os *splines* multilineares contínuos, pois as restrições de grau e continuidade estão automaticamente asseguradas. A base minimal, quando extraída a partir de grades regulares, é similar a esquemas de diferenças finitas com interpolação linear [Cri77]. A base maximal, por outro lado, apresenta um caráter multiescala. Encerramos o capítulo descrevendo dois algoritmos eficientes que encontram respectivamente $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ com tempo de execução máximo da ordem do número de células folha das grades diádicas a partir das quais foram extraídas.

Nos capítulos seguintes, descrevemos uma série de aplicações para *splines* diádicos. No capítulo 4 consideramos a aproximação de funções pelo método de mínimos quadrados. Neste capítulo também apresentamos um algoritmo que realiza o refinamento de uma grade diádica G a partir de uma função f qualquer, tornando G mais segmentada nas regiões do domínio onde f possui um comportamento não linear mais acentuado. Mostramos por experimentos que o uso de grades diádicas adaptativas proporciona resultados tão precisos quanto o de uma grade uniforme de mesma resolução máxima, a uma fração do custo.

No capítulo 5 revisamos a técnica de Galerkin para a solução de equações diferenciais parciais. Ilustramos essa técnica resolvendo um problema linear, a

equação de Helmholtz [Gom99]. No capítulo 6, estendemos essa técnica a equações diferenciais variáveis no tempo. Para autenticarmos essa nova abordagem, simulamos um processo de difusão do calor em meio isotrópico e comparamos os resultados com dados obtidos analiticamente.

Nos capítulos 7 a 9 enfocamos o problema da simulação do escoamento de petróleo. No capítulo 7 introduzimos as equações de escoamento multifásico óleo/água que, no capítulo 8, discretizamos e adaptamos à modelagem por elementos finitos. O capítulo 9 descreve os módulos do programa de simulação que desenvolvemos (Simóleo), e apresenta alguns testes realizados para avaliá-lo. Concluímos, no capítulo 10, com observações principalmente a respeito destes testes e outras considerações finais.

1.4 Revisão bibliográfica

1.4.1 *Splines* diádicos

Existe uma grande variedade de trabalhos baseados em *splines* na literatura ([Gom99, BBB87] por exemplo). Em geometria computacional e outras áreas, as grades diádicas são conhecidas como *k-d trees* [HS99]. Duchaineau [Duc96] estudou especificamente os *splines* diádicos, e mostrou que eles podem ser usados em esquemas de aproximação, compactação e projeto interativo de funções.

1.4.2 Simulação de escoamento multifásico

No tocante à simulação do escoamento de petróleo, diversos experimentos encorajam o uso de refinamento local. Esta técnica vem tradicionalmente sendo utilizada com modelagens por diferenças finitas, método que emprega grades uniformes e portanto dificulta abordagens dinâmicas e com muitos níveis de refinamento. A maioria das propostas nesse sentido envolvem uma malha grosseira global e algumas malhas mais finas nas regiões críticas do domínio. As iterações numéricas são realizadas alternadamente entre a malha grosseira e as malhas finas, sendo que os resultados de cada uma gera as condições de contorno para a outra. Além disso, esquemas especiais são necessários para traduzir as propriedades entre os níveis de refinamento. As pressões, normalmente representadas por funções suaves, são obtidas por meio de interpolação. As saturações, por outro lado, são dadas por funções mais irregulares e portanto são estimadas geralmente por meio de replicação de valores. Citamos a seguir alguns destes trabalhos.

Heinemann *et al* [HGH83] desenvolveram um simulador que permite a realização de refinamento local dinâmico de malhas tridimensionais com um limite de 8 divisões das células iniciais em cada direção. Este simulador foi criado para dar suporte a várias formulações, como esquemas *Black-Oil* e fluxos de vapor, que podem ser avaliadas como processos simultâneos. Os escoamentos baseados no modelo *Black-Oil* usam equações no formato IMPES (*Implicit Pressure Explicit Saturation* [Cri77]), com passo de tempo também variável.

Estes pesquisadores concluíram que o refinamento local é capaz de reduzir o tempo de execução do sistema se o número de células empregadas nas malhas pode ser reduzido em 20%.

Wasserman [Was87] sugeriu uma técnica na qual o refinamento local é aplicado em *janelas*, que são regiões retangulares com até três dimensões. Em seu trabalho as janelas são refinadas estaticamente, e a partir das propriedades do reservatório em suas fronteiras são calculados os valores correspondentes em cada uma de suas sub-regiões. Nestes cálculos, as pressões são obtidas por interpolação, e as saturações por meio de replicação.

A solução dos sistemas de equações resultantes é feita de maneira que a adição de novas janelas ao modelo aumenta linearmente, e não exponencialmente, seu número de operações. Exemplos de simulação mostraram que o refinamento vertical é importante.

Nacul *et al* [NLP+90] mostraram algumas formas de acelerar a convergência de sistemas com refinamento local associados à técnica de decomposição de domínios. Esta técnica consiste em dividir a ampla região do reservatório em subdomínios e resolver os sistemas numéricos referentes a cada um deles separadamente.

O uso de pré-condicionadores, ou seja, a aplicação de resultados obtidos por simulações sobre malhas grosseiras como valores iniciais dos métodos iterativos reduziu o tempo de convergência de todos os experimentos realizados. A quantidade de iterações também foi reduzida fazendo um tipo de relaxação nas variáveis dos sistemas. Analisando os casos nos quais os subdomínios possuíam regiões em comum, Nacul *et al* verificaram que quando estas regiões são pequenas as taxas de convergência podem ser melhoradas, ao passo que regiões comuns muito grandes em geral aumentam o tempo de processamento.

Duas técnicas foram investigadas por Risso [Ris02] com a finalidade de reduzir o tempo das simulações: o refinamento local e as fronteiras abertas. Para realizar esta investigação, construiu um modelo base contendo certas regiões críticas próximas aos poços. Risso dividiu seus experimentos com refinamento local em

três etapas, nas quais realizou testes com refinamentos graduais e abruptos, restritos ou além das regiões de interesse e com grades de várias resoluções, com ou sem refinamento das regiões críticas em cada uma delas. Após um estudo comparativo, verificou que o refinamento local, além das regiões críticas, é indesejável porque reduz pouco o erro em relação ao aumento no tempo de simulação causado pelo uso de grades mais finas. Os menores tempos de simulação foram obtidos usando a malha grosseira, que entretanto mostrou-se inviável em função da baixa qualidade dos resultados. A melhor escala de refinamento tanto do modelo base quanto das regiões críticas foi de 1:4, com consideráveis ganhos nos tempos de execução. Risso concluiu que o refinamento local é mais indicado para reservatórios pequenos com poucos poços, e sugeriu que as regiões críticas de reservatórios muito grandes devem ser estudadas como reservatórios individuais, juntamente com uma avaliação cuidadosa dos fluxos em suas fronteiras.

Capítulo 2

Grades Adaptativas

Neste capítulo descrevemos as grades diádicas, que permitem a discretização heterogênea dos domínios de integração. Através desta heterogeneidade podemos melhorar a precisão das soluções obtidas, refinando as bases de aproximação onde estas soluções possuem comportamento mais irregular. A estrutura simples das grades diádicas também facilita a representação e manipulação destas estruturas em computadores.

2.1 Grades diádicas

Podemos obter uma *grade diádica* através de divisões sucessivas de uma célula inicial ou *célula raiz*, correspondente à totalidade do domínio, sempre pela metade. Cada nova divisão é feita por um plano perpendicular a um eixo de coordenadas diferente, numa ordem cíclica. Por exemplo, para um domínio tridimensional com eixos cartesianos x, y e z, se a primeira divisão for perpendicular a x, então a segunda deverá ser perpendicular ao eixo y, a terceira perpendicular a z, a quarta perpendicular a x, e assim por diante, ciclicamente (figuras 2.1 e 2.2). Esta organização pode ser aplicada a um número arbitrário de dimensões $k \geq 1$, e faz com que o refinamento da grade ocorra gradativamente. Por fim, obtemos um conjunto sub-regiões retangulares.

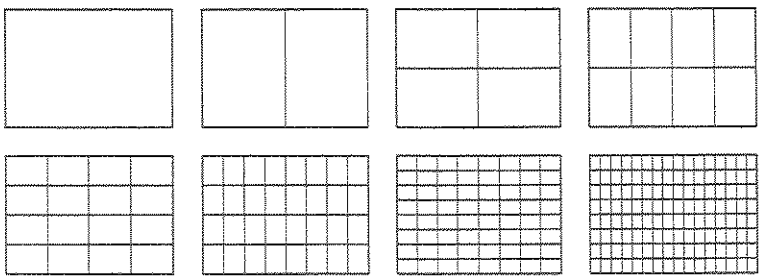


Figura 2.1: Criação de uma grade diádica bidimensional.

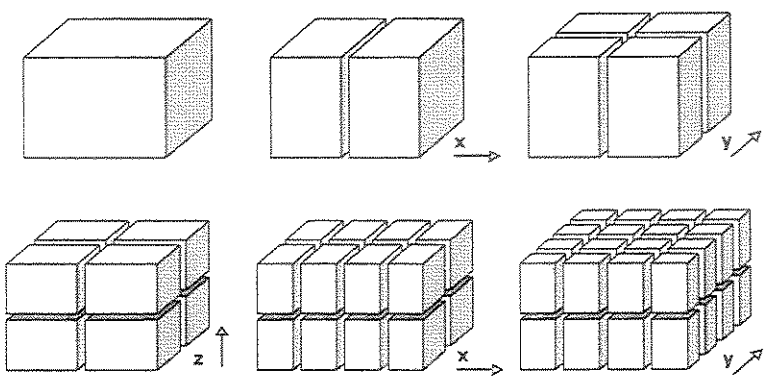


Figura 2.2: Criação de uma grade diádica tridimensional.

Assim, cada célula pode ser dividida em duas *células filhas* (figura 2.3) e, a menos que seja a célula raiz, equivale à metade de sua *célula mãe*.

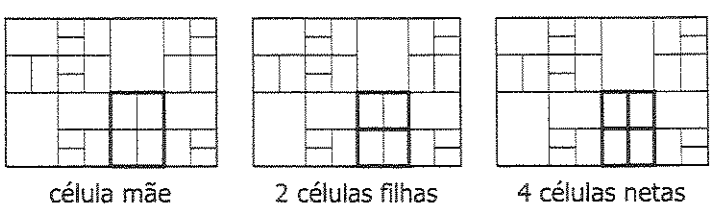


Figura 2.3: Uma célula mãe, com duas células filhas e quatro células netas.

Prolongando-se esta disposição indefinidamente obtemos a *grade diádica infinita* G^*. Na prática usamos uma *grade diádica finita*, em que o processo termina depois de um número finito de divisões e portanto todas as grades mencionadas no restante deste trabalho devem ser consideradas finitas.

As *células folha* são o conjunto de células da grade que não sofrem divisões. Na figura 2.3, entre as células destacadas, a célula mãe e as células filhas não são folhas, ao contrário das células netas.

Logicamente, grades diádicas correspondem a *árvores binárias* (figura 2.4), e modelar uma grade diádica equivale a escolher uma árvore binária correspondente limitando adequadamente sua profundidade em cada região do domínio.

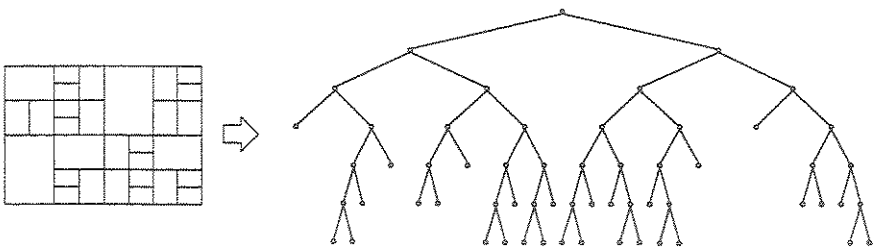


Figura 2.4: Árvore binária correspondente a uma grade.

2.2 Orientação, indexação e profundidade

Para simplificar os cálculos, vamos supor que a origem do sistema de coordenadas é um vértice da célula raiz, e que todas as arestas são paralelas aos eixos coordenados (figura 2.5).

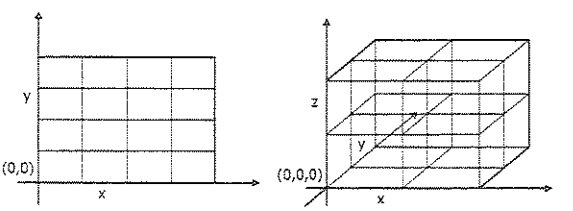


Figura 2.5: A origem coincide com um dos vértices da célula raiz.

Atribuímos a cada célula da grade diádica um *índice*, que é um valor inteiro positivo. A célula raiz, a única que não provem da partição de outras, tem índice 1. Uma célula filha de uma célula de índice I recebe índice $2I$ ou $2I+1$, conforme esteja respectivamente mais próxima ou distante da origem. Assim, podemos representar cada célula de maneira compactada com apenas 1 bit. Veja a distribuição dos índices em uma grade bidimensional na figura 2.6.

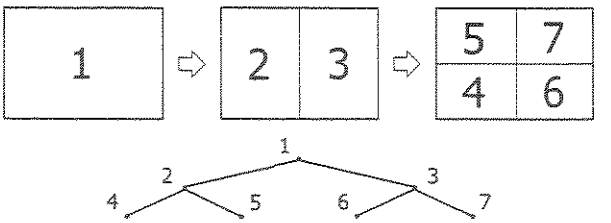


Figura 2.6: Distribuição dos índices em uma grade 2d.

Toda célula também tem uma *profundidade* na árvore binária derivada da grade diádica: a célula raiz tem profundidade 0, e cada filha de uma célula de profundidade P tem profundidade $P+1$. Ou seja, a profundidade de uma célula C indica o número de cortes necessários para transformar a célula raiz na célula C. O *nível* P da grade é o conjunto de todas as células com profundidade P, que pode ter no máximo 2^P células. A figura 2.1 contém os primeiros 8 níveis (do zero ao sétimo) da grade diádica infinita G^*. Na figura 2.7 as células de profundidade 5 (o nível 5) estão em destaque.

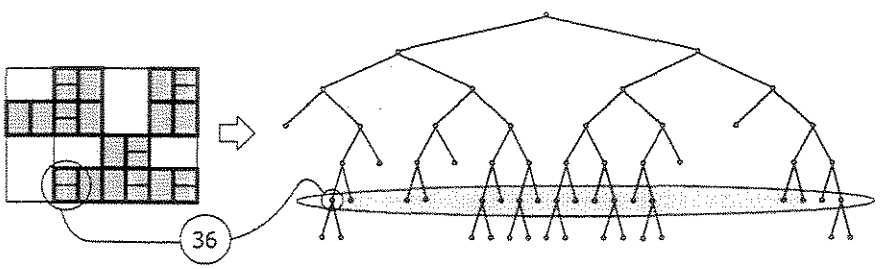


Figura 2.7: Nível 5 de uma grade diádica.

2.3 Posicionamento e tamanho das células

Cada nível de profundidade da grade infinita G^* funciona como uma espécie de tabuleiro regular, com espaçamentos iguais ao longo de cada eixo, sempre em quantidades equivalentes a alguma potência negativa de 2 vezes a dimensão correspondente da célula raiz. Portanto a posição e tamanho de uma célula podem ser obtidos facilmente a partir de sua profundidade P e índice I.

Como a profundidade reflete a quantidade de cortes necessários para gerar uma determinada célula, e lembrando que estes cortes ocorrem alternando os eixos ciclicamente, então podemos calcular a quantidade máxima de células Q_i

que podem ser dispostas ao longo do eixo i de uma grade G com k dimensões, a um nível de profundidade P:

$$q_i = \left\lfloor \frac{P}{k} \right\rfloor + \begin{cases} 1 & \text{se} \quad P \bmod k > i \\ 0 & \text{se} \quad P \bmod k \leq i \end{cases}$$

$$Q_i = 2^{q_i}$$

Para encontrarmos, por exemplo, o vetor Q das possíveis posições distintas de uma célula com profundidade $P = 5$ em G, sendo $k = 3$, fazemos:

$$\begin{aligned} i = 0 \quad &\text{(eixo X)}, \quad Q_0 = 2^{\lfloor \frac{5}{3} \rfloor + 1} = 4 \quad (5 \bmod 3 > 0) \\ i = 1 \quad &\text{(eixo Y)}, \quad Q_1 = 2^{\lfloor \frac{5}{3} \rfloor + 1} = 4 \quad (5 \bmod 3 > 1) \\ i = 2 \quad &\text{(eixo Z)}, \quad Q_2 = 2^{\lfloor \frac{5}{3} \rfloor + 0} = 2 \quad (5 \bmod 3 \leq 2) \end{aligned}$$

A *posição* de uma célula C é o vetor $\delta(C) = (\delta_0, \ldots, \delta_{k-1})$ que dá o número de células de profundidade P entre C e a origem em cada eixo. Os valores $\delta_0, \delta_1, \ldots, \delta_{k-1}$ podem ser extraídos do índice I de C, bastando apenas desentrelaçar seus P bits menos significativos, organizando-os em k grupos. Mais precisamente, a representação binária de cada δ_i é composta pelo i-ésimo bit menos significativo de I somado aos demais bits tomados de k em k, até a última casa antes do bit 1 mais significativo.

Assim, por exemplo, se a grade G for dividida até que todas as suas células alcancem profundidade 5, com um total de $2^5 = 32$ células, então cada eixo terá respectivamente $Q = (4, 4, 2)$ células. Se uma célula C tiver índice $I = 57$, de representação binária $\overset{*01201}{111001}_2$, então sua posição é obtida separando-se os 5 bits menos significativos como $\delta(C) = \left(\overset{00}{10}_2, \overset{11}{11}_2, \overset{2}{0}_2\right) = (2, 3, 0)$. Observe a estrutura da grade G e a posição da célula C na figura 2.8.

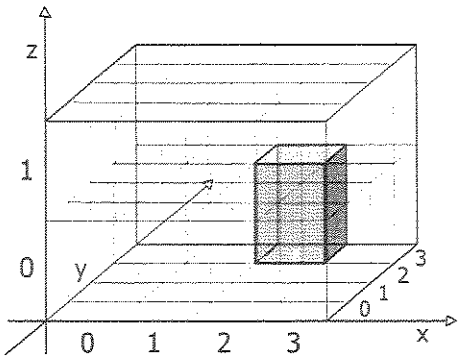


Figura 2.8: Célula C, com vetores $Q(C) = (4, 4, 2)$ e $\delta(C) = (2, 3, 0)$.

2.4 Geometria canônica

Numa grade diádica, como a divisão das células segue sempre a mesma ordem cíclica de eixos, a escolha das proporções da célula raiz interfere diretamente na velocidade com que os refinamentos trazem ganhos de precisão numérica. Isto ocorre porque os erros de aproximação de espaços de *splines* são proporcionais ao diâmetro das células, isto é, às suas maiores arestas. Portanto, uma célula isométrica (com todas as arestas iguais) precisa ser dividida ao menos k vezes, em uma grade de dimensão k, para que o erro atribuído às células resultantes seja reduzido significativamente. Além disso a divisão produziria 2^k células isométricas e consequentemente este problema se repetiria em todos os níveis.

Para que as células em todos os níveis de profundidade mantenham-se proporcionais, a célula raiz deve ter no eixo i tamanho $2^{-i/k}$, com i variando entre 0 e $k-1$. Uma célula raiz bidimensional assim definida terá lados de tamanho $1 : 2^{-1/2}$ ou $1 : \sqrt{0,5}$. Em 3 dimensões, estas proporções tornam-se $1 : 2^{-1/3} : 2^{-2/3}$. Todas as células de profundidade P serão similares à célula raiz, submetida a uma escala linear de fator $2^{-P/k}$, e rodada por uma permutação de eixos.

Desta forma, o diâmetro das células é reduzido por um fator constante de $2^{-1/k}$ a cada nível. Embora as células apresentem maior precisão em eixos diferentes de nível para nível, esta abordagem leva a refinamentos graduais da grade diádica, e é isotrópica se considerarmos todos os níveis em conjunto.

Capítulo 3

Splines diádicos

Mostramos em detalhes neste capítulo a família das funções *tenda*, um subespaço multilinear do espaço de *splines* gerais definidos sobre grades diádicas. Destacamos duas bases específicas para estas funções multilineares, e mostramos algoritmos discretos para encontrá-las além de um algoritmo simples capaz de interpolar uma função qualquer usando funções tendas. Analisando os algoritmos que encontram as bases, mostramos que possuem um tempo de execução proporcional ao número de células folha das grades diádicas sobre as quais são chamados.

3.1 *Splines*

Um *spline* é uma função polinomial por partes, ou seja, uma função f sobre cujo domínio Ω existe uma partição (*grade*) em regiões distintas (*células*), sendo que f possui comportamento polinomial em cada célula. Quando o domínio Ω está estruturado como uma grade diádica G, f é um *spline diádico*. A função f restrita a alguma destas células do domínio é chamada de *retalho polinomial* (ou simplesmente *retalho*). A continuidade mínima de f nas fronteiras entre todos os retalhos indica a *continuidade* de f. O *grau máximo* de f é dado pelo grau máximo de todas as variáveis independentes sobre todos seus retalhos. A vantagem do uso de *splines* está na reconhecida simplicidade dos polinômios combinada com a possibilidade de melhorar a qualidade da aproximação numa região arbitrária de Ω, usando uma partição mais fina nesta região.

3.2 Elementos finitos

O *suporte* de um *spline* é o conjunto de células onde ele não é identicamente nulo. Assim, outra vantagem de *splines* como funções aproximadoras é que seus espaços muitas vezes possuem bases cujos elementos (funções das bases) têm suporte muito pequeno, independente do número total de células na grade (figura 3.1).

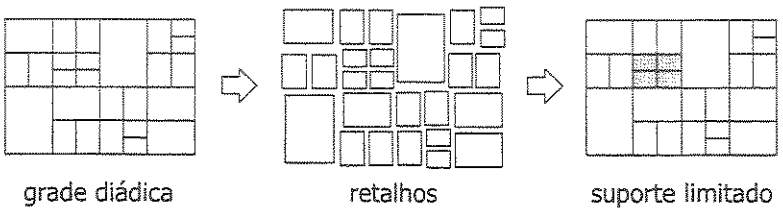


Figura 3.1: Suporte de um *spline* de aproximação.

Splines com esta característica são chamados *elementos finitos*. Para obtermos o valor de uma função $u^*(x) = \sum a_i \varphi_i(x)$ num dado ponto x combinando elementos finitos de uma base φ, só é necessário calcularmos os termos $\varphi_i(x)$ tais que x pertença ao suporte de φ_i, o que normalmente ocorre com poucas funções da base. Na aplicação de métodos numéricos, como o método de Galerkin ou aproximações por mínimos quadrados, os sistemas de equações montados com estas bases normalmente são representáveis por matrizes esparsas. A escolha de alguma base de aproximação depende do grau de suavidade buscado. *Splines* de graus elevados podem gerar aproximações mais fiéis a funções suaves, usando menos retalhos.

As funções aproximadoras que escolhemos neste trabalho são elementos finitos multilineares, ou seja, *splines* multilineares definidos sobre uma grade diádica finita G de dimensão k.

3.3 *splines* multilineares sobre grades diádicas

Um *spline* multilinear de domínio k-dimensional definido sobre uma grade diádica G possui um retalho sobre cada célula de G, tem continuidade 0 e grau máximo 1 em suas k variáveis. Embora paralelamente a cada eixo a variação de um *spline* multilinear seja linear, em outras direções a variação pode ser quadrática ou cúbica, em decorrência do produto de suas k variáveis (grau total k).

Chamaremos de $\mathcal{E}^1[G]$ o espaço das funções multilineares de fronteira nula definido sobre uma grade G diádica. Modificando a estrutura da grade G, alteramos em geral o espaço $\mathcal{E}^1[G]$ e portanto suas bases.

3.4 As funções *tenda*

Nas bases que consideramos neste trabalho, o suporte de um elemento k-dimensional τ é composto por 2^k células de mesmo tamanho e profundidade, agrupadas em torno de um *vértice central* v. Neste ponto o elemento da base, τ, possui valor 1, caindo linearmente ao longo de cada eixo para 0 na fronteira do suporte. No caso particular de $k = 2$, o gráfico do elemento genérico assemelha-se a uma tenda e por isso os elementos (para qualquer k) serão aqui designados *funções tenda* ou simplesmente *tendas* (figura 3.2).

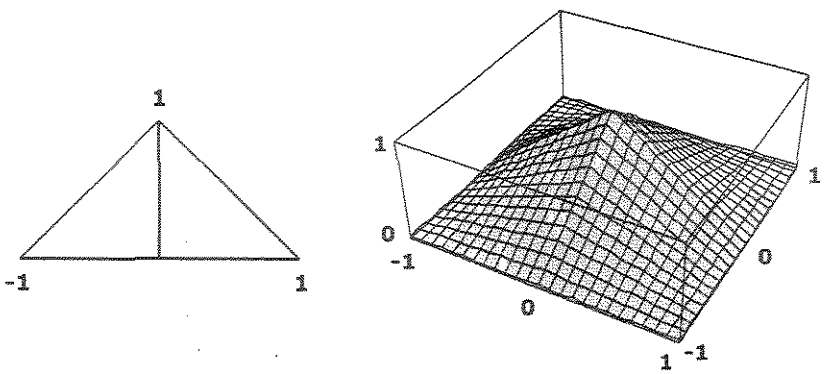


Figura 3.2: Função tenda com uma e duas dimensões.

A profundidade (posto) de uma tenda é a profundidade das células de seu suporte. Uma tenda pode ser identificada unicamente por sua *célula superior* C, a célula de seu suporte mais distante da origem. Denotaremos por τ_C a tenda de uma grade diádica G cuja célula superior é C, e podemos escrevê-la como

$$\tau_C(x_0, \ldots, x_{k-1}) = \prod_{i=0}^{k-1} \Lambda\left(2^{i/k} Q_i x_i - \delta_i\right) \tag{3.1}$$

onde Q_i é o número de células de profundidade P na grade infinita G^* ao longo da direção i; δ_i é a componente i do vetor posição δ de C; e Λ é a função definida por

$$\Lambda(x) = \max\{0, 1 - |x|\} = \begin{cases} 1 - x & \text{se } 0 \leq x \leq 1 \\ 0 & \text{se } |x| > 1 \\ 1 + x & \text{se } -1 \leq x < 0 \end{cases} \tag{3.2}$$

É importante observar que esta definição da tenda τ_C admite que a grade G mede $2^{-i/k}$ em cada direção i. Se ao longo de cada eixo a grade G variasse apenas de zero a um, a fórmula de τ_C seria $\prod_{i=0}^{k-1} \Lambda\left(Q_i x_i - \delta_i\right)$.

3.5 As bases $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$

Utilizamos para cada espaço de *splines* $\mathcal{E}^1[G]$ duas bases, $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$. Em ambas há uma tenda para cada *vértice completo* de G, ou seja, para cada vértice que tem 2^k células incidentes. A base $\mathcal{B}_{max}$ (*maximal*) consiste nas tendas de maior suporte possível (figura 3.4), enquanto $\mathcal{B}_{min}$ (*minimal*) consiste nas tendas de menor suporte possível (figura 3.3). Quando G é uma grade regular de profundidade P, $\mathcal{B}_{max}$ possui tendas em todos os níveis de k até P, e $\mathcal{B}_{min}$ possui tendas apenas de nível P.

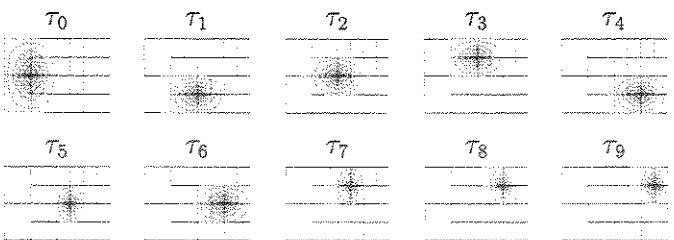


Figura 3.3: A base minimal do espaço $\mathcal{E}^1$ de uma grade bidimensional irregular G.

Note que nas figuras 3.3 e 3.4 as tendas numeradas iguais possuem o mesmo vértice central, e a diferença entre as duas bases está nas tendas τ_2, τ_6 e τ_8.

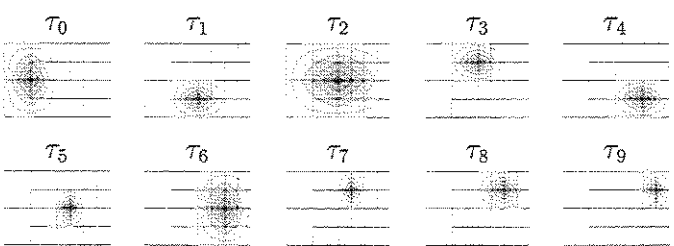


Figura 3.4: A base maximal do espaço $\mathcal{E}^1$ de uma grade bidimensional irregular G.

A bases $\mathcal{B}_{max}$ podem ser encontradas mais facilmente do que as bases $\mathcal{B}_{min}$, mas possuem como desvantagem numérica uma tendência a gerar matrizes densas do produto escalar de suas funções. Por outro lado, como suas tendas são mais abrangentes, permitem realizar aproximações grosseiras de amplas regiões do domínio utilizando apenas subconjuntos delas.

As bases $\mathcal{B}_{min}$ têm como propriedade gerar as matrizes de produtos escalares mais esparsas possíveis, pois as regiões comuns ao suporte de suas tendas são minimizadas. Isso aumenta a estabilidade numérica porque reduz ao máximo a interferência no sistema de alterações na base.

3.5.1 Independência linear das bases $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$

Provaremos que $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ são de fato bases para um espaço $\mathcal{E}^1[G]$ mostrando que as funções tenda destes conjuntos são *linearmente independentes (LI)* e *geram* o espaço $\mathcal{E}^1[G]$. Começaremos mostrando a independência linear de todo *conjunto simples*, ou todo conjunto de tendas que possui no máximo uma tenda para cada vértice.

Lema 3.1 *Seja $\mathcal{B}$ um conjunto simples de tendas, e τ uma tenda de profundidade mínima em $\mathcal{B}$ com vértice central v. Qualquer combinação linear do conjunto $\mathcal{B}' = \mathcal{B} - \tau$ é nula em v.*

Prova: Para qualquer tenda τ' de $\mathcal{B}'$ o ponto v está situado na fronteira ou no exterior do suporte de τ', e é zero neste ponto. □

Teorema 3.1 *Qualquer conjunto simples $\mathcal{B}$ de tendas é linearmente independente.*

Prova: Suponhamos que $\mathcal{B}$ não seja LI, ou seja, existe alguma tenda de $\mathcal{B}$ que pode ser escrita como combinação linear das demais. Neste caso, seja τ uma tenda nestas condições com profundidade P mínima em $\mathcal{B}$ e vértice central v. Obviamente ela pode ser escrita como combinação linear das outras tendas, com profundidades maiores ou iguais à P. Contudo, pelo lema 3.1 isto não é possível porque qualquer combinação linear das outras tendas de $\mathcal{B}$ terá valor zero em v, onde τ é um e portanto $\mathcal{B}$ só pode ser LI. □

3.5.2 Interpolação usando tendas

Antes de prosseguir com as provas, mostraremos um algoritmo simples de interpolação usando tendas. Seja G uma grade diádica k-dimensional, $\mathcal{B}$ um conjunto qualquer de tendas e f uma função arbitrária definida sobre G. O algoritmo 3.5.1

algoritmo 3.5.1 Interpola($\mathcal{B}$, f)

1: Seja V o conjunto dos vértices centrais das tendas de $\mathcal{B}$;
2: Ordene $\mathcal{B}$ em ordem de profundidade, obtendo $\tau_1, \ldots, \tau_n$;
3: **for** $i : 1 \to n$ **do**
4: Seja v o vértice central de τ_i;
5: **if** $v \notin V$ **then**
6: $a_i \leftarrow 0$;
7: **else**
8: $a_i \leftarrow f(v) - \sum\{a_j\tau_j(v) : j < i \text{ e } v \in \mathrm{Dom}(\tau_j)\}$;
9: $V \leftarrow V - \{v\}$;
10: **end if**
11: **end for**

(Interpola) encontra uma combinação linear $g = \sum_{i=0}^{n} a_i\tau_i$ de $\mathcal{B}$ que interpola f nos vértices centrais das tendas de $\mathcal{B}$.

Lema 3.2 *A função g resultante do algoritmo 3.5.1 tem $g(v) = f(v)$ em todo vértice central v das tendas do conjunto $\mathcal{B}$.*

Prova: No passo 8, cada coeficiente a_i é calculado de forma que $f(v) = \sum_j a_j\tau_j(v)$, para $j \leq i$ (a restrição $v \in \mathrm{Dom}(\tau_j)$ é supérfula pois $\tau_j(v) = 0$ se $v \notin \mathrm{Dom}(\tau_j)$), onde v é o vértice central de τ_i. Verifica-se que a combinação linear $\sum_j a_j\tau_j(v)$, com $j > i$ das demais tendas de $\mathcal{B}$ é sempre nula. Isto ocorre porque as tendas com $j > i$ ou têm centro em v, e portanto terão $a_j = 0$ (pelos passos 5 e 9), ou têm centro diferente de v e profundidades maiores ou iguais à profundidade de τ_i, e portanto $\tau_j(v) = 0$ pelo lema 3.1. □

3.5.3 Cobertura das bases $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$

Vamos mostrar agora que as bases $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ de uma grade k-dimensional G geram todo o espaço $\mathcal{E}^1[G]$. Na verdade provaremos que qualquer conjunto de tendas que inclui pelo menos uma tenda para cada vértice completo de G (um *conjunto completo* de tendas) gera todo o espaço $\mathcal{E}^1[G]$.

Diremos que um vértice v é *esquina* de uma face F de G se v for face (de dimensão 0) de F. Se v está sobre alguma face F (com dimensão $d \leq k$) de alguma célula folha C de G mas não é esquina de F, então v é *intermediário* de F e de C. Diremos que v é *incompleto* se não está na fronteira da grade e é intermediário de alguma célula folha.

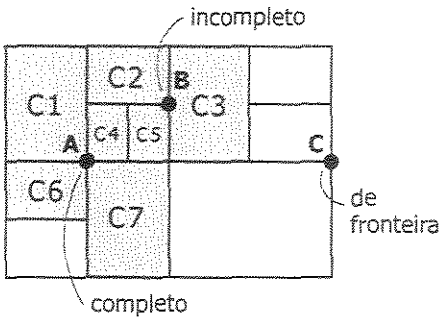


Figura 3.5: Um vértice completo A, esquina das células C1, C4, C6 e C7; um vértice incompleto B, esquina das células C2, C5 e intermediário à célula C3; e um vértice de fronteira C.

Em uma grade diádica G, para todo vértice v seja $N(v)$ o conjunto das células folha incidentes a v. Se v é completo então ele é esquina de todas as células de $N(v)$. Quando v é incompleto, ele sempre é esquina de pelo menos duas células de $N(v)$ e intermediário de pelo menos uma destas células.

Lema 3.3 *Seja G uma grade diádica k-dimensional, f uma função de $\mathcal{E}^1[G]$, e v um vértice intermediário de uma face F de alguma célula folha C. Se f é nula em todos os vértices de F, é nula em v.*

Prova: Não pode haver mais do que um retalho polinomial de f em C, porque C é uma célula folha. Este retalho é multilinear dentro de G. Como f é contínua, então a restrição $f|F$ também deve ser uma função multilinear. Assim, a variação de f entre quaisquer dois vértices adjacentes de F é linear e consequentemente f terá valor 0 sobre toda a aresta entre estes dois pontos. Repetindo este argumento em todos os eixos, concluímos que f é zero em toda face F, e portanto no vértice v. □

Lema 3.4 *Seja v um vértice incompleto. Se v é intermediário de uma célula C, então C tem profundidade menor que a profundidade de qualquer célula da qual v é esquina.*

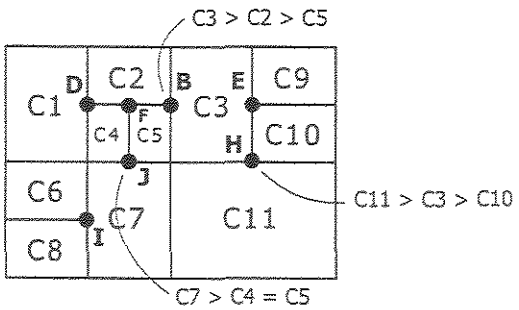


Figura 3.6: Cada vértice incompleto (B, D, E, F, H, I e J) é intermediário a uma célula de profundidade menor do que todas as células das quais são esquinas.

Prova: Seja G uma grade diádica k-dimensional. Se v é um vértice completo de G e C é a célula folha de menor profundidade da qual v é esquina, então v também é esquina de outras $2^k - 1$ células (não necessariamente folhas) de G de mesma profundidade. Agora seja v um vértice incompleto (como ilustrado na figura 3.6) e $N(v)$ o conjunto das células folha incidentes em v. Seja C_1 a célula de menor profundidade P_1 em $N(v)$ que tem v como esquina e C_2 uma célula de $N(v)$ com profundidade P_2, da qual v é intermediário. Refinemos G até encontrar uma grade G', onde todas as células têm profundidade mínima P_1. O vértice v será completo em G' e esquina de 2^k células de profundidade P_1. Como G' é um refinamento de G, algumas destas células serão descendentes de C_2 e portanto $P_2 < P_1$. □

Lema 3.5 *Seja G uma grade diádica k-dimensional e f uma função do espaço $\mathcal{E}^1[G]$. Se f é zero em todos os vértices completos de G então é zero também em todos os vértices incompletos de G.*

Prova: Seja H o grafo orientado cujos nós são os vértices de G, e no qual cada vértice incompleto v aponta para as esquinas de todas as faces das quais ele é intermediário. Note que se f é zero nessas esquinas, então pelo lema 3.3, f é zero também em v. Denotaremos por $\kappa(v)$ a profundidade da maior célula de quem v é esquina. Este grafo não pode ter ciclos, porque para qualquer aresta de v_1 para v_2, pelo lema 3.4 $\kappa(v_1) > \kappa(v_2)$. Os sorvedouros de H são vértices completos ou de fronteira. Como f é nula nestes pontos, então (por indução na distância ao sorvedouro mais distante) f é nula em todos os vértices de G. □

Teorema 3.2 *Todo conjunto completo de tendas $\mathcal{B}$ definido sobre uma grade diádica G gera o espaço $\mathcal{E}^1[G]$.*

Prova: Sejam f um *spline* de $\mathcal{E}^1[G]$ e $\mathcal{B}$ um conjunto completo de tendas sobre G. Pelo lema 3.2 podemos encontrar uma combinação linear g de $\mathcal{B}$ tal que $g(v) = f(v)$ em todos os vértices completos de G. Vamos mostrar que $g = f$. Como as tendas de $\mathcal{B}$ pertencem ao espaço $\mathcal{E}^1[G]$, a diferença $r = g - f$ também pertence a este espaço; ela é zero na fronteira de G e é zero em todos os vértices completos de G. Pelo lema 3.5, r também é zero em todos os vértices incompletos de G e, pelo lema 3.3, r é identicamente nula em G. □

As bases $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ são conjuntos simples e completos, de maneira que pelos teoremas 3.1 e 3.2 são de fato bases do espaço $\mathcal{E}^1[G]$.

3.6 Determinação de bases $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$

É importante que tenhamos algoritmos eficientes para encontrar as bases $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ de uma grade diádica G, porque qualquer alteração em G normalmente modifica estas bases. Como há uma correspondência entre tendas e vértices completos, então obteremos $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ em princípio encontrando os vértices completos de G. Se para cada vértice v encontrado guardarmos uma tenda τ cujo suporte é composto pelas células de menor profundidade que têm v como esquina, chegaremos a $\mathcal{B}_{max}$. Fazendo o mesmo, mas escolhendo as células de maior profundidade, chegaremos a $\mathcal{B}_{min}$.

3.6.1 Lugares e o conjunto estrela

Antes de mostrarmos os algoritmos que encontram as bases, precisaremos de duas definições úteis para trabalharmos com sub-regiões de grades diádicas, os *lugares* e suas *estrelas*. Um *lugar* L de uma grade diádica G k-dimensional consiste de uma profundidade $P(L)$, e uma face $F(L)$ de alguma célula de nível $P(L)$ da grade G que não está na fronteira de G. Dizemos que a *dimensão* do lugar L é a dimensão d de $F(L)$. Na grade diádica infinita G^* há exatamente 2^{k-d} células de profundidade $P(L)$ incidentes à face $F(L)$. Se todas estas células também são células da grade G (não necessariamente folhas) então L é dito um *lugar completo* de G. Veja a figura 3.7 abaixo. A lista destas células é a *estrela* de L, denotada por L^*.

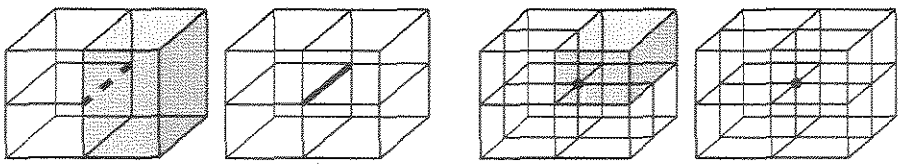


Figura 3.7: Lugares incompletos cujas faces são respectivamente um segmento de reta e um ponto, na primeira e terceira grade a partir da posição mais à esquerda. Na segunda e quarta grade, as células em destaque são divididas para que estes lugares tornem-se completos.

Note que uma mesma face F pode ser parte de dois lugares distintos, com níveis diferentes (figura 3.8).

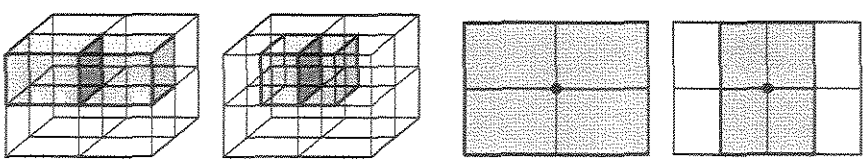


Figura 3.8: Nas duas grades tridimensionais, uma mesma face F plana determinada por células de profundidade 3 ou 4. Na grade bidimensional, uma face F (ponto) determinada por células de profundidade 2 ou 3.

Se $F(L)$ é uma célula, então L é sempre completo e L^* tem uma única célula, a própria $F(L)$. Numa grade bidimensional, a estrela de um lugar aresta consiste de duas células (figura 3.9, esquerda), enquanto em uma grade tridimensional a estrela de um lugar aresta tem 4 células (figura 3.9, direita). Os lugares completos de dimensão zero são os vértices completos de G.

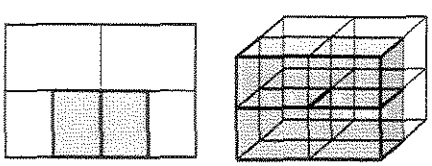


Figura 3.9: Um lugar aresta em uma grade bidimensional e em uma grade tridimensional.

3.6.2 O algoritmo Gera_Bmax

Gera_Bmax é um procedimento recursivo que encontra a base $\mathcal{B}_{max}$ de uma grade G de dimensão k, fazendo um percurso em profundidade nos lugares de G em busca

de vértices completos. As entradas do algoritmo são uma sequência $\langle I_0, \ldots, I_{n-1} \rangle$ de índices que representam as células da estrela L^* de um lugar completo L de G; a dimensão d deste lugar; o eixo de divisão $i = P(L) \bmod k$; e a lista $H = \langle h_0, \ldots, h_{m-1} \rangle$ dos eixos ortogonais a $F(L)$, onde $m = k - d$. O algoritmo devolve a lista dos índices das células superiores das tendas da base. Para todo $r \in 0, \ldots, m-1$ e todo $j \in 0, \ldots, n-1$, o bit 2^r do índice j diz se a célula I_j está mais perto (0) ou mais longe (1) da origem do que a face $F(L)$ na direção do eixo h_r.

algoritmo 3.6.1 Gera_Bmax($\langle I_0, \ldots, I_{n-1} \rangle$, d, i, $\langle h_0, \ldots, h_{m-1} \rangle$)

```
1: if d = 0 then                                          L é um ponto.
2:   Devolve {I_{n-1}};          Inclui a nova tenda (B_max ← B_max ∪ {I_{n-1}}).
3: else
4:   if ⟨I_0, ..., I_{n-1}⟩ não possui folhas then
5:     Calcule A = ⟨I'_0, ..., I'_{n-1}⟩, tal que I'_j = (2I_j);
6:     Calcule B = ⟨I''_0, ..., I''_{n-1}⟩, tal que I''_j = (2I_j + 1);
7:     l ← (i + 1) mod k;
8:     if i = h_r para algum r em 0, ..., m − 1 then
9:       Calcule C ← {C_j = A_j se o bit r de j é 1, ou C_j = B_j se este bit é 0 };
10:      B' ← Gera_Bmax(C, d, l, H);
11:      Devolve B';
12:    else
13:      B' ← Gera_Bmax(A, d, l, H);
14:      B'' ← Gera_Bmax(B, d, l, H);
15:      B''' ← Gera_Bmax(A · B, d − 1, l, H · ⟨i⟩);
16:      Devolve B' ∪ B'' ∪ B''';
17:    end if
18:  else
19:    Devolve {};
20:  end if
21: end if
```

A chamada inicial é Gera_Bmax($\langle 1 \rangle$, k, 0, $\langle\rangle$).

Lema 3.6 *O algoritmo Gera_Bmax executado sobre um lugar completo L encontra todos os lugares completos de dimensão zero (1) cujo vértice está em $F(L)$ e (2) cuja profundidade P é a menor possível mas $P \geq P(L)$.*

Prova: Precisamos mostrar que o algoritmo Gera_Bmax encontra todos os vértices completos (e consequentemente as tendas) existentes no interior da face $F(L)$, e

que as células do suporte destas tendas são escolhidas de forma que elas têm a menor profundidade possível em $F(L)$. Seja V o conjunto de todos estes vértices completos.

Primeiro o algoritmo verifica a dimensão d, na linha 1. Se esta for zero então L é um vértice; como L é completo, ele determina uma tenda cujo suporte é L^*, com vértice central $F(L) = v$. Neste caso $V = \{v\}$, $m = k$, e o algoritmo devolve a célula I_{n-1}, que é a célula de L^* incidente a v mais distante da origem em todas as direções (linha 2).

Na linha 4, se alguma célula da lista $\langle I_0, \ldots, I_{n-1} \rangle$ é folha, então, como L tem dimensão $d > 0$, não há vértices completos em $F(L)$ e portanto o conjunto V é vazio (linha 19). Caso contrário, as células de L^* podem ser todas divididas ao longo do eixo i (linhas 5 e 6), criando os conjuntos A e B.

Essa divisão pode acompanhar uma divisão de L ou não, conforme a posição do eixo i em relação a $F(L)$. Quando i é ortogonal a $F(L)$ (linha 8, figura 3.10), a face $F(L)$ não é modificada, mas L é substituído por um lugar L' tal que $F(L') = F(L)$ e $P(L') = P(L) + 1$. Cada célula C_j de L'^* é uma das filhas (A_j ou B_j) da célula correspondente de L^*. Especificamente, suponha que $i = h_r$; então C_j é A_j se o bit r de j é 1, senão C_j é B_j (linha 9). Como $F(L') = F(L)$, a lista B' de índices devolvida na linha 10 representa tendas com os mesmos vértices centrais V.

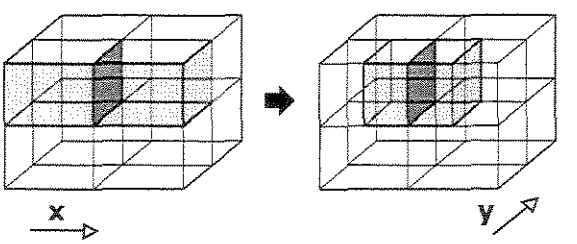


Figura 3.10: Na grade à esquerda, o eixo de divisão (x) é ortogonal à face em destaque.

Quando o eixo i é paralelo a $F(L)$ (linha 12), os índices de A e B representam respectivamente L'^* e L''^*, as estrelas dos lugares L' e L'' resultantes da divisão de L ao meio. A concatenação de A e B, por sua vez, representa L'''^*, a estrela do lugar com dimensão $d - 1$ que separa L' e L'' (figura 3.11). Neste caso três instâncias de Gera_Bmax são chamadas, nas linhas 13, 14 e 15, para procurarem os vértices completos V', V'', e V''' sobre $F(L')$, $F(L'')$ e $F(L''')$. Como estas faces são disjuntas e sua união é $F(L)$, então $V' \cap V'' \cap V''' = \{\}$ e $V = V' \cup V'' \cup V'''$.

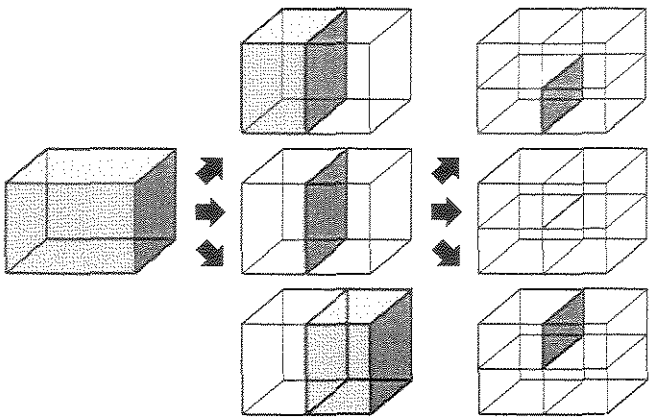


Figura 3.11: Gera_Bmax executado sobre duas faces determinadas por células cujos eixos de divisão são paralelos a estas faces.

Para todo vértice $v \in V$, a profundidade de qualquer lugar que tem v como face será pelo menos $P(L)$ (se $F(L)$ for um ponto) e portanto a condição (2) do lema será respeitada. □

Teorema 3.3 *O algoritmo 3.6.1 (Gera_Bmax) executado sobre uma grade diádica G k-dimensional encontra a base $\mathcal{B}_{max}$ de G.*

Prova: Executando Gera_Bmax sobre o lugar L, cuja estrela L^* contém apenas a célula raiz de G, encontraremos pelo lema 3.6 todos os vértices completos no domínio de G, $(F(L))$, além de um conjunto de índices representando as tendas de maiores suportes em G com centro nestes vértices. □

3.6.3 O algoritmo Gera_Bmin

O algoritmo Gera_Bmin é uma modificação de Gera_Bmax, com uma condição adicional no comando if na linha 1:

if $d = 0$ e $\langle I_0, \ldots, I_{n-1} \rangle$ possui alguma folha **then**

...

end if

Lema 3.7 *O algoritmo Gera_Bmin executado sobre um lugar completo L encontra todos os lugares completos de dimensão zero (1) cujo vértice está em $F(L)$ e (2) cuja profundidade P é a maior possível.*

Prova: A demonstração é análoga à do lema 3.6. A única diferença é que no passo 1, se a condição é satisfeita, obviamente L é o lugar de *maior* profundidade centrado no vértice $v = F(L)$.

Se $d = 0$ mas L^* não possui folhas, então $F(L)$ será um vértice v ortogonal a todos os k eixos de coordenadas e o algoritmo será chamado recursivamente (linhas 9 e 10) sobre as células filhas de L^* que têm v como esquina, até chegar a um lugar L', onde L'^* tem alguma célula folha. Neste caso, a tenda τ cujo suporte é L'^* tem profundidade máxima em $F(L)$. □

Teorema 3.4 *O algoritmo (Gera_Bmin) executado sobre uma grade diádica G k-dimensional encontra a base $\mathcal{B}_{min}$ de G.*

Prova: Também executando Gera_Bmin sobre o lugar L, cuja estrela L^* contém apenas a célula raiz de G, encontraremos pelo lema 3.7 todos os vértices completos no domínio de G, $(F(L))$, e um conjunto de índices representando as tendas de menores suportes em G com centro nestes vértices. □

3.6.4 A complexidade de Gera_Bmax e Gera_Bmin

Calcularemos o tempo de execução dos algoritmos Gera_Bmax e Gera_Bmin, capazes de encontrar respectivamente as bases $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ de uma grade diádica G de dimensão k, em função do número n de células folha de G. Este número reflete a complexidade da grade G, independente da forma com que ela foi refinada.

Toda grade diádica é o resultado de uma série de divisões de sua célula raiz. Cada divisão acrescenta exatamente duas células à grade, o que em sua estrutura de árvore binária significa transformar uma célula folha na mãe de duas novas células folha. Da mesma forma, o número de células folha é incrementado de uma unidade. Assim, se G tem n células folha, então foi gerada a partir de $n - 1$ divisões de sua célula raiz, e possui $(2 \times n) - 1$ células ao total.

Suponhamos que G é bidimensional, e seja C uma célula de G. A célula C pode, no máximo, pertencer à estrela de 4 lugares de dimensão 0 + 4 lugares de dimensão 1 + um lugar de dimensão 2 = 9 lugares distintos. Como Gera_Bmin

verifica cada lugar de G uma única vez, então não pode ter um tempo de execução superior a $9 \times ((2 \times n) - 1)$, ou seja, sua complexidade é da ordem de $\Theta(n)$ quando executado sobre grades bidimensionais.

Se C é uma célula de uma grade G tridimensional, uma verificação análoga nos indica que no máximo C pode fazer parte da estrela de 27 lugares distintos. Portanto, como G também possui $(2 \times n) - 1$ células neste caso, então Gera_Bmin também tem complexidade da ordem de $\Theta(n)$ quando executado sobre grades tridimensionais. Considerando que o algoritmo Gera_Bmax verifica uma quantidade de lugares menor ou igual ao número total de lugares de G, então possui complexidade máxima igual à de Gera_Bmin.

Capítulo 4

Aproximação por mínimos quadrados

A aproximação por mínimos quadrados é uma técnica bastante conhecida. O uso de tendas nestes casos pode ser vantajoso, considerando a estrutura simples destas funções, de fácil armazenamento e manipulação. Além disso, na aproximação de funções muito irregulares, podemos refinar as bases até alcançarmos o grau de precisão requerido, sem desperdício de recursos.

4.1 Aproximação por mínimos quadrados

Seja $u(x)$ uma função que queremos aproximar, definida sobre um domínio Ω de dimensão k. Buscamos uma aproximação que seja uma combinação linear $u^*(x) = \sum_{j=0}^{n-1} a_j \varphi_j(x)$, onde φ é um conjunto linearmente independente de funções definidas sobre Ω. Seja $\mathcal{S}^*$ o espaço gerado pela base $\langle \varphi_0, \ldots, \varphi_{n-1} \rangle$. Segundo o critério dos *mínimos quadrados*, a melhor aproximação para u no espaço $\mathcal{S}^*$ é a que minimiza a distância

$$\|u - u^*\| = \left(\int_\Omega (u(x) - u^*(x))^2 dx \right)^{1/2} \tag{4.1}$$

onde $dx = dx_{k-1} \ldots dx_0$. A equação (4.1) pode ser escrita também como

$$\|u - u^*\| = \langle u - u^*, u - u^* \rangle^{1/2}$$

onde $\langle , \rangle$ denota o *produto escalar* de duas funções sobre Ω, definido por

$$\langle f, g \rangle = \int_\Omega f(x) g(x) dx \tag{4.2}$$

Por brevidade, omitiremos o argumento "(x)" nas integrais a seguir.

Verifica-se que minimizar (4.1) equivale a tornar u^* a projeção ortogonal de u sobre o espaço de aproximação $\mathcal{S}^*$, ou seja, resolver as equações

$$\langle u - u^*, \varphi_i \rangle = 0 \quad \equiv \quad \int_\Omega (u - u^*)\varphi_i dx = 0 \qquad i = 0, \ldots, n-1 \tag{4.3}$$

Ou seja,

$$\int_\Omega (u - \sum_{j=0}^{n-1} a_j \varphi_j)\varphi_i dx = 0 \qquad i = 0, \ldots, n-1 \tag{4.4}$$

Desenvolvendo o lado direito desta equação para um dado i, obtemos

$$\int_\Omega u\,\varphi_i dx - \int_\Omega (\sum_{j=0}^{n-1} a_j \varphi_j)\varphi_i dx = \int_\Omega u\,\varphi_i dx - \sum_{j=0}^{n-1} a_j \int_\Omega \varphi_j \varphi_i dx$$

de maneira que podemos reescrever as equações (4.4) como

$$\sum_{j=0}^{n-1} a_j \int_\Omega \varphi_j \varphi_i dx = \int_\Omega u\,\varphi_i dx \qquad i = 0, \ldots, n-1 \tag{4.5}$$

Esta fórmula é o Sistema Normal de n equações lineares sobre n variáveis, os coeficientes $a_0, \ldots, a_{n-1}$. Sua representação matricial tem a forma $Ea = b$, onde E é uma matriz $n \times n$, e b é um vetor coluna de tamanho n, definidos por

$$E_{ij} = \langle \varphi_i, \varphi_j \rangle \qquad b_i = \langle u, \varphi_i \rangle \qquad i, j = 0, \ldots, n-1$$

4.2 O produto escalar de funções tenda

Nesta seção mostramos como calcular o produto escalar $\langle \varphi_i, \varphi_j \rangle$ de duas funções tenda, necessário à composição da matriz E do sistema (4.5). Na seção 4.3 mostraremos uma fórmula genérica para o cálculo aproximado de um elemento $b_i = \langle u, \varphi_i \rangle$, para uma função qualquer f.

Sejam τ' e τ'' duas tendas definidas sobre uma grade diádica G de dimensão k. Obviamente, para calcular o produto escalar $\langle \tau', \tau'' \rangle$ precisamos calcular a integral (4.2) apenas na região R (figura 4.1) coberta pelos suportes das duas tendas.

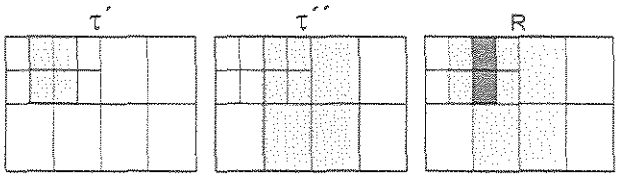


Figura 4.1: Tendas τ', τ'', e a região R comum aos suportes destas funções.

Supondo sem perda de generalidade que o posto P' de τ' é maior ou igual ao posto P'' de τ'', verifica-se que esta região é a união de no máximo 2^k células $C'_0, \ldots, C'_{m-1}$ de posto P', contidas no suporte de τ'. Portanto, podemos escrever

$$\langle \tau', \tau'' \rangle = \int_\Omega \tau'\tau'' dx = \sum_{j=0}^{m-1} \int_{C'_j} \tau'\tau'' dx \tag{4.6}$$

No restante desta seção veremos como calcular a integral $\int_{C'} \tau'\tau'' dx$, para uma célula $C' = C'_j$.

Pela definição (3.1),

$$\tau'(x) = \prod_{i=0}^{k-1} \Lambda(2^{i/k} Q'_i x_i - \delta'_i) \qquad e \qquad \tau''(x) = \prod_{i=0}^{k-1} \Lambda(2^{i/k} Q''_i x_i - \delta''_i) \tag{4.7}$$

Seja C'' a célula do suporte de τ'' que contém a célula C' (figura 4.2).

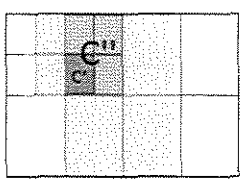


Figura 4.2: A célula C' de τ' está contida na célula C'' de τ''.

Para x em C', podemos reescrever a fórmula da tenda τ'' como

$$\tau''(x) = \prod_{i=0}^{k-1} \Lambda(2^{i/k} Q''_i x_i - \delta''_i) = \prod_{i=0}^{k-1} \Lambda(\alpha_i (2^{i/k} Q'_i x_i - \delta'_i) + \beta_i)$$

Nestas fórmulas, α_i é o tamanho do suporte de τ' em relação ao suporte de τ'' na direção i, e $\beta_i = (2^{i/k} Q''_i v'_i - \delta''_i)$ onde v' é o vértice central do suporte de τ'.

Portanto, a parcela do produto escalar $\langle \tau', \tau'' \rangle$ correspondente a C' é a integral

$$\int_{x_0^o}^{x_0^f} \cdots \int_{x_{k-1}^o}^{x_{k-1}^f} \prod_{i=0}^{k-1} \Lambda(2^{i/k} Q_i' x_i - \delta_i') \prod_{i=0}^{k-1} \Lambda(\alpha_i (2^{i/k} Q_i' x_i - \delta_i') + \beta_i) dx_{k-1} \ldots dx_0 \quad (4.8)$$

onde x_i^o e x_i^f são respectivamente as posições de C' mais próxima e mais distante da origem ao longo do eixo i. Uma vez que cada fator da produtória depende de uma única coordenada x_i, podemos separar as integrais, ou seja, usar a identidade

$$\int_{x^o}^{x^f} \int_{y^o}^{y^f} f(x) g(y) dy dx = \left(\int_{x^o}^{x^f} f(x) dx \right) \left(\int_{y^o}^{y^f} g(y) dy \right)$$

e reordenar a expressão (4.8) como segue:

$$\prod_{i=0}^{k-1} \int_{x_i^o}^{x_i^f} \Lambda(2^{i/k} Q_i' x_i - \delta_i') \Lambda(\alpha_i (2^{i/k} Q_i' x_i - \delta_i') + \beta_i) dx_i \quad (4.9)$$

Para simplificar, fazemos as seguintes mudanças de variáveis

$$u_i = 2^{i/k} Q_i' x_i - \delta_i' \qquad \text{portanto} \qquad dx_i = (2^{i/k} Q_i')^{-1} du_i \quad (4.10)$$

e transformamos a equação (4.9) em

$$\prod_{i=0}^{k-1} (2^{i/k} Q_i')^{-1} \int_{u_i^o}^{u_i^f} \Lambda(u_i) \Lambda(\alpha_i u_i + \beta_i) du_i$$

Se C' for uma célula inferior de τ' na direção i, isto é, se ao longo do eixo i a célula C' estiver entre o vértice central v' e a origem, os limites de integração $[u_i^o, u_i^f]$ nesta direção serão $[-1, 0]$ e $\Lambda(u_i)$ será $1 + u_i$. Por outro lado, se C' for superior na direção i, isto é, se v' estiver entre C' e a origem nesta direção, estes limites de integração serão $[0, 1]$ e $\Lambda(u_i)$ será $1 - u_i$. Analogamente, se C'' for uma célula inferior de τ'' na direção i, então $\Lambda(\alpha_i u_i + \beta_i) = 1 + \alpha_i u_i + \beta_i$; caso contrário, temos que $\Lambda(\alpha_i u_i + \beta_i) = 1 - \alpha_i u_i - \beta_i$. Portanto, a integral no fator i da produtória pode ser desenvolvida de quatro meneiras:

- C' e C'' inferiores,

$$\int_{-1}^{0} (1 + u_i)(1 + \alpha_i u_i + \beta_i) du_i = \left(-\frac{\alpha_i}{6} + \frac{\beta_i}{2} + \frac{1}{2} \right) \quad (4.11)$$

- C' inferior e C'' superior,

$$\int_{-1}^{0}(1+u_i)(1-\alpha_i u_i-\beta_i)du_i=\left(\frac{\alpha_i}{6}-\frac{\beta_i}{2}+\frac{1}{2}\right) \tag{4.12}$$

- C' superior e C'' inferior,

$$\int_{0}^{1}(1-u_i)(1+\alpha_i u_i+\beta_i)du_i=\left(\frac{\alpha_i}{6}+\frac{\beta_i}{2}+\frac{1}{2}\right) \tag{4.13}$$

- C' e C'' superiores,

$$\int_{0}^{1}(1-u_i)(1-\alpha_i u_i-\beta_i)du_i=\left(-\frac{\alpha_i}{6}-\frac{\beta_i}{2}+\frac{1}{2}\right) \tag{4.14}$$

4.3 Produto escalar de funções gerais

Para calcular um elemento do vetor b do sistema (4.5) precisamos calcular a integral $\int_\Omega u\,\varphi dx$ onde u é uma função arbitrária. Normalmente esta integral não possui uma fórmula explícita. Portanto, devemos usar um método de integração numérica para calcular uma aproximação da mesma.

4.3.1 Quadratura de Gauss

Uma boa escolha neste sentido é a *quadratura de Gauss*, que em uma dimensão é a aproximação

$$\int_{-1}^{+1} f(t)dt \simeq \sum_{j=0}^{n} C_j f(t_j) \tag{4.15}$$

para certos pontos $t_j \in [-1,+1]$ e certos coeficientes C_j. Os pontos t_j são as raízes de algum polinômio de grau $n+1$ de uma família H de polinômios ortogonais no intervalo $[-1,+1]$, e os coeficientes C_j são

$$C_j=\int_{-1}^{+1} l_j(x)dx,$$

onde $l_0,\ldots,l_n$ são os *polinômios de Lagrange* sobre esses pontos [Pis80]. Escolhendo para H os *polinômios de Legendre*, podemos usar valores já tabelados de C_j e t_j. Quanto maior o n, melhor será a precisão do resultado. Em particular, se f for um polinômio de grau $2n+1$, o resultado da integração é exato.

Os pontos e coeficientes da fórmula de quadratura de Gauss-Legendre são tabelados para o intervalo de integração $[-1, 1]$. Para calcular a integral num intervalo genérico $[a, b]$,

$$\int_a^b f(x)dx$$

precisamos fazer a substituição de variável

$$x = \frac{b-a}{2}t + \frac{b+a}{2}$$

de modo que se $x = a$, $t = -1$, e $x = b$, $t = +1$. Temos então

$$\int_a^b f(x)dx = \int_{-1}^{1} \frac{b-a}{2} f(\frac{b-a}{2}t + \frac{b+a}{2})dt$$

e portanto a fórmula (4.15) fica

$$\int_a^b f(x)dx = \frac{b-a}{2} \sum_{j=0}^{n} C_j f(\frac{b-a}{2}t_j + \frac{b+a}{2}) \tag{4.16}$$

Para $n = 10$, usamos os seguintes valores de t_j e C_j:

t_j	C_j
-0,973906528517172	0,066671344308688
-0,865063366688985	0,149451349150581
-0,679409568299024	0,219086362515982
-0,433395394129247	0,269266719309996
-0,148874338981631	0,295524224714753
0,148874338981631	0,295524224714753
0,433395394129247	0,269266719309996
0,679409568299024	0,219086362515982
0,865063366688985	0,149451349150581
0,973906528517172	0,066671344308688

Tabela 4.1: Valores de t_j e C_j, para $n = 10$.

4.3.2 Integração em k variáveis

Se f é uma função de k variáveis, e Ω é uma caixa com projeções $[-1, +1]$ em cada eixo i, então a fórmula de quadratura de Gauss fica

$$\int_\Omega f(x)dx \simeq \sum_{j_0=0}^{n} \sum_{j_1=0}^{n} \cdots \sum_{j_{k-1}=0}^{n} C_{j_0} C_{j_1} \ldots C_{j_{k-1}} f(t_{j_0} t_{j_1} \ldots t_{j_{k-1}}) \tag{4.17}$$

Ou seja, a função f é calculada numa grade com $(n+1)^k$ pontos cujas coordenadas são todas as combinações $t_{j_0}, t_{j_1}, \ldots, t_{j_{k-1}}$ dos pontos $t_0, t_1, \ldots, t_{k-1}$ da fórmula (4.15); e os valores obtidos são somados com pesos correspondentes $C_{j_0}, C_{j_1}, \ldots, C_{j_{k-1}}$. Quando o domínio Ω é uma caixa geral, com projeção $[a, b]$ em cada eixo i, a fórmula (4.17) deve ser ajustada conforme a fórmula (4.16).

4.3.3 Produtos escalares

A integral $\int_\Omega u\,\varphi\,dx$, onde $\varphi(x)$ é uma função da base e $u(x)$ é uma função geral, pode ser então aproximada aplicando-se a quadratura de Gauss à função $f = u \times \varphi$. No caso particular em que φ é uma tenda τ, é preferível calcular a integral separadamente sobre cada uma das células do suporte de $\tau(x)$.

4.4 Exemplos usando grades regulares

Para testar a qualidade dos espaços de *splines* diádicos $\mathcal{E}^1[G]$ como espaços aproximantes, aplicamos o método dos mínimos quadrados para as funções da tabela 4.2 usando várias grades regulares G. Uma vez que todas as funções do espaço $\mathcal{E}^1[G]$ são nulas na fronteira da célula raiz, escolhemos para teste funções com esta mesma característica.

nome	fórmula
sxsy	$\sin(2\pi x)\sin(2\sqrt{2}\pi y)$
gauss	$exp(-((x-0,5)^2+(\sqrt{2}y-0,5)^2)/0,005)$
tendaR	$\Lambda(2x-1)\Lambda(2\sqrt{2}y-1)$

Tabela 4.2: Funções para testes do método de mínimos quadrados.

Utilizamos as bases de tendas maximais ($\mathcal{B}_{max}$) e minimais ($\mathcal{B}_{min}$) obtidas a partir dos níveis 6, 8 e 10 da grade diádica bidimensional infinita. Para avaliarmos numericamente cada aproximação, calculamos a distância $\|u - u^*\|$ pela fórmula (4.1) entre as funções originais e as funções aproximadas. Neste cálculo, a integral (4.1) foi obtida por quadratura de Gauss (seção 4.3.1) aplicada separadamente para cada célula da grade G. Os resultados destes cálculos são mostrados na tabela 4.3. Esta tabela serve tanto para a base $\mathcal{B}_{max}$ quanto para a base $\mathcal{B}_{min}$, pois os resultados obtidos foram praticamente idênticos para ambas as bases.

nível	6	8	10
λ	0,1531	0,0765	0,0383
sxsy	0,01468220	0,00347907	0,00085685
gauss	0,01386992	0,00297978	0,00068360
tendaR	$6{,}3\times10^{-17}$	$6{,}8\times10^{-17}$	$6{,}0\times10^{-17}$

Tabela 4.3: Distâncias $\|u - u^*\|$ caculadas para cada base de aproximação. O parâmetro λ é o diâmetro das células da grade.

As figuras 4.3 a 4.5 ilustram estas funções, suas aproximações nas três grades e os respectivos resíduos $u - u^*$. Cada função f está representada em curvas de nível. O parâmetro δ_1 é o espaçamento entre os níveis, sendo o primeiro nível $\binom{+}{-}\delta_1/2$. O parâmetro δ_2 é o último nível traçado. A região entre os níveis $-\delta_1/2$ e $+\delta_1/2$ não é pintada; tons vermelhos indicam valores positivos e tons azuis indicam valores negativos. Os valores de δ_1 e δ_2 estão indicados nas legendas de cada figura. Todos os gráficos de funções em curvas de nível no restante deste trabalho devem ser interpretadas da mesma forma.

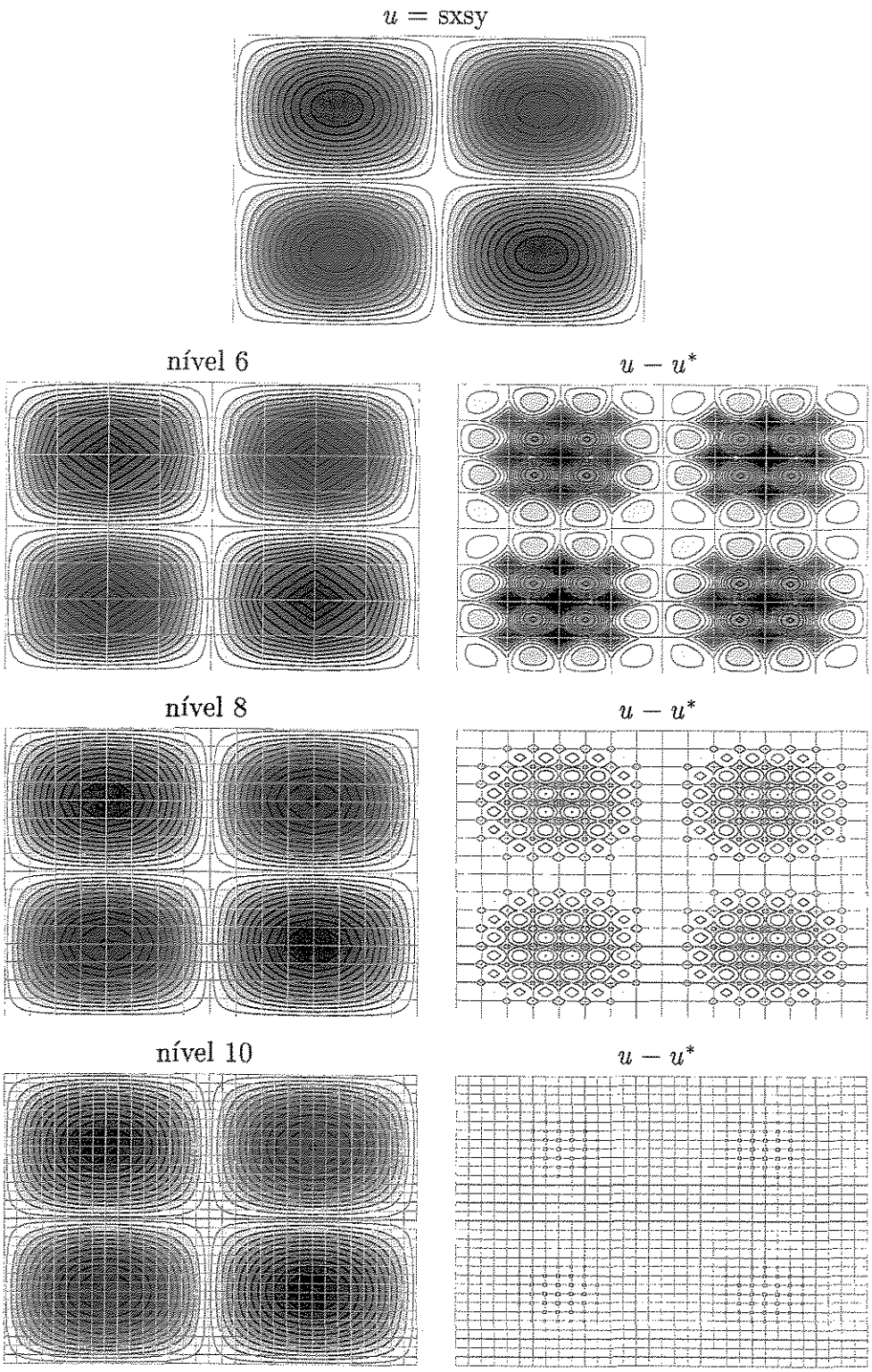


Figura 4.3: A função *sxsy* e suas aproximações por mínimos quadrados nas bases $\mathcal{B}_{min}$ de níveis 6, 8, e 10, traçadas com $\delta_1 = 0,08$ e $\delta_2 = 1,2$. Os resíduos $u - u^*$ foram traçados com $\delta_1 = 0,008$ e $\delta_2 = 0,12$.

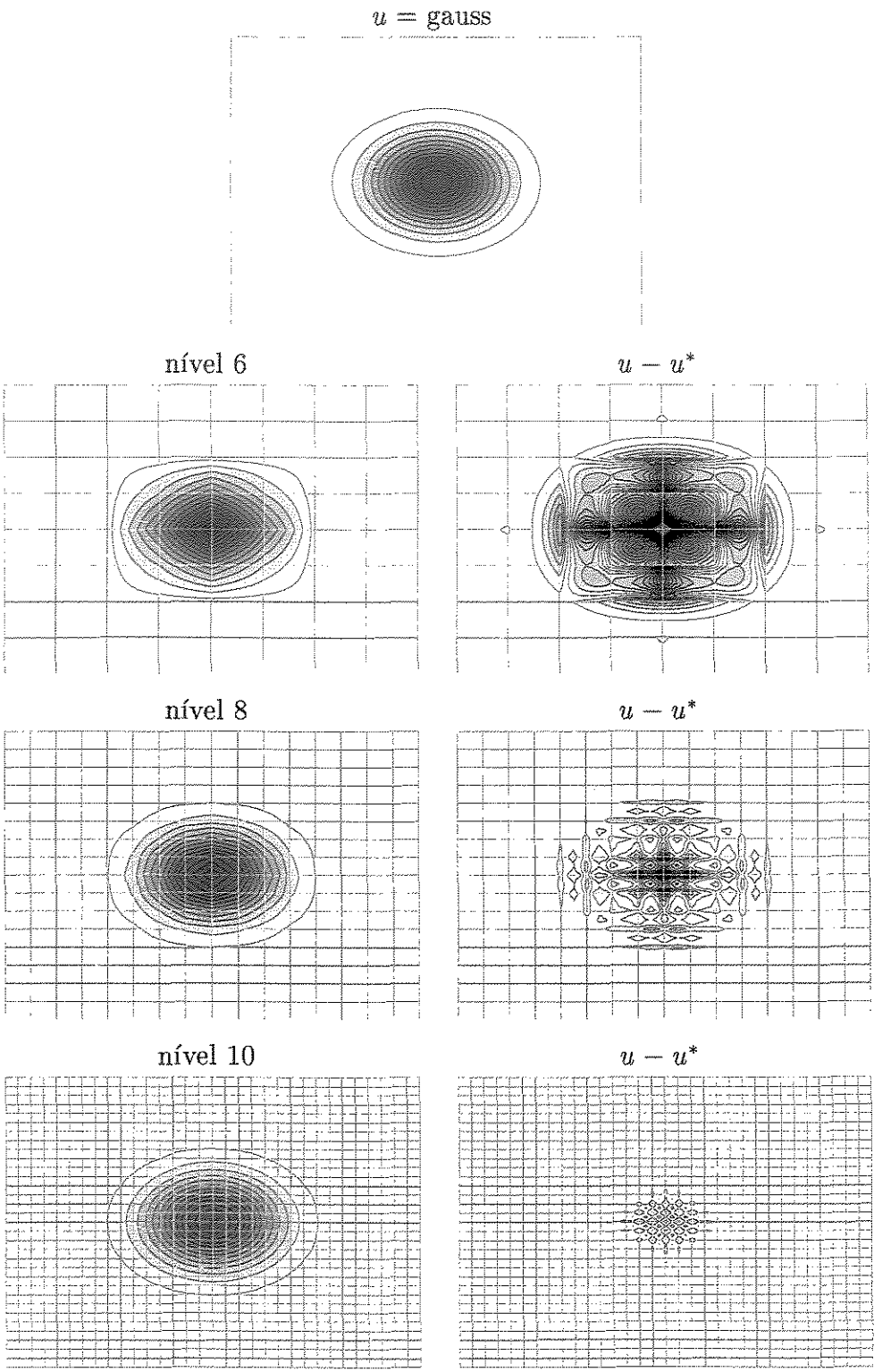


Figura 4.4: Função *gauss* e aproximações nas bases $\mathcal{B}_{min}$ de níveis 6, 8, e 10, com $\delta_1 = 0,08$ e $\delta_2 = 1,2$. Para os resíduos $u - u^*$, foram usados $\delta_1 = 0,008$ e $\delta_2 = 0,12$.

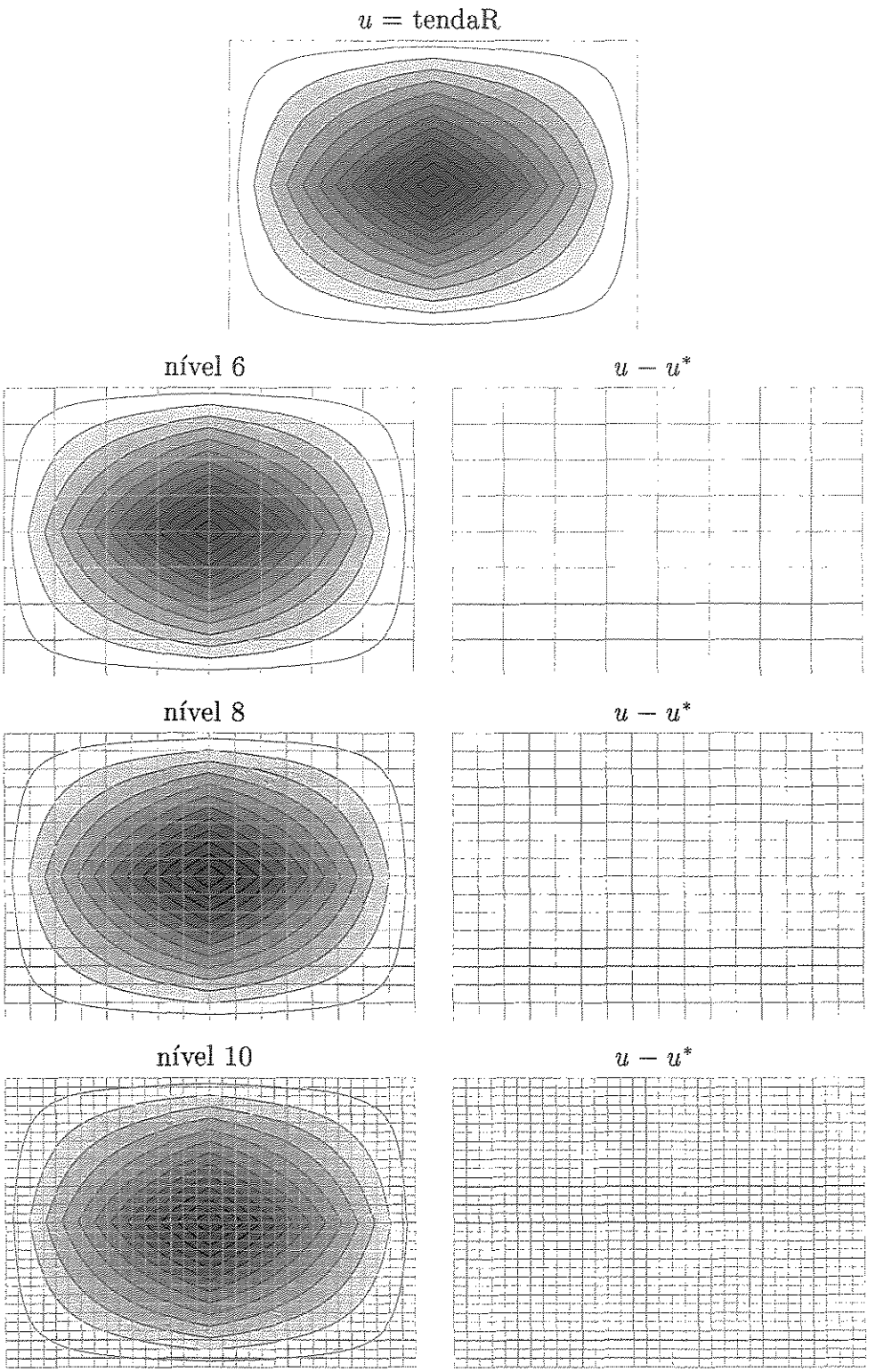


Figura 4.5: Função *tendaR* e aproximações nas bases $\mathcal{B}_{min}$ de níveis 6, 8, e 10 com $\delta_1 = 0,08$ e $\delta_2 = 1,2$. Os resíduos $u - u^*$ possuem $\delta_1 = 0,008$ e $\delta_2 = 0,12$.

Como previsto, todas as aproximações da função *tendaR* (que pertence ao espaço $\mathcal{E}^1[G]$, apesar de não estar na base $\mathcal{B}_{min}$) são praticamente iguais à função original; os resíduos são desprezíveis e refletem apenas erros de arredondamento. No caso das funções *sxsy* e *gauss*, o erro é aproximadamente proporcional ao quadrado do diâmetro λ das células, como esperado pela teoria para aproximações de grau um.

4.5 Aproximações com refinamento da grade

Os gráficos do resíduo $u - u^*$ evidenciam que os erros de aproximação normalmente concentram-se onde as funções originais possuem maior curvatura ao longo dos eixos. Interessa portanto escolher uma grade irregular G mais refinada onde é mais difícil simular o comportamento de u usando tendas. Isto permite obter uma aproximação u^* bastante precisa, com uma base $\mathcal{B}$ de dimensão muito menor.

Para fins de teste, usamos grades construídas pelo algoritmo recursivo 4.5.1 (*Refina_Grade*). Suas entradas são a função u a ser aproximada; a dimensão k; o índice I e profundidade P de uma célula C; um parâmetro *lim*; e a profundidade máxima P_{max}. O algoritmo divide a célula C e suas descendentes até que o seguinte critério de divisão não seja mais satisfeito. Chamamos de *canto inferior* o ponto r de C mais próximo da origem e de *canto superior* o ponto s de C mais distante da origem. A *face inferior* de C na direção i é o subconjunto dos pontos da face $F(C)$ com coordenada $r[i]$ no eixo i (com dimensão $k-1$) e a *face superior* de C na direção i é a face análoga. O critério de divisão consiste em verificar se a diferença máxima max_var entre o valor de u no ponto central de C (linha 5) e as interpolações dos valores de u nos pontos centrais das faces inferiores e superiores em cada direção (linhas 6 a 19) é maior do que o parâmetro *lim*, respeitando o limite de profundidade P_{max} (linha 20).

O tempo de execução de Refina_Grade é obviamente proporcional ao tamanho da árvore construída, que é proporcional ao número de células folha da mesma. No pior caso, este algoritmo retorna uma grade G uniforme, onde todas as células folha têm profundidade P_{max}, com $2^{P_{max}}$ células folha, $2 \times 2^{P_{max}} - 1$ células no total (seção 3.6.4), com tempo de execução da ordem de $\Theta(2^{P_{max}})$.

algoritmo 4.5.1 Refina_Grade(u, k, I, P, lim, P_{max})

```
1: C ← célula de índice I;
2: ctr ← centro de (C);
3: r ← canto inferior de (C);
4: s ← canto superior de (C);
5: max_var ← 0; val_ctr ← u(ctr);
6: for i : 1 → k do
7:    for j : 1 → k do
8:       if (j ≠ i) then
9:          p'[j] ← ctr[j]; p''[j] ← ctr[j];
10:      else
11:         p'[i] ← r[i]; p''[i] ← s[i];
12:      end if
13:   end for
14:   val_est ← (u(p') + u(p''))/2;
15:   δ ← |val_ctr − val_est|;
16:   if (δ > max_var) then
17:      max_var ← δ;
18:   end if
19: end for
20: if (max_var > lim) and (P < P_max) then
21:   D ← Refina_Grade(u, k, 2I, P + 1, lim, P_max);
22:   E ← Refina_Grade(u, k, 2I + 1, P + 1, lim, P_max);
23:   devolva uma árvore G com sub-árvores D e E;
24: else
25:   devolva uma árvore G com uma única célula folha I;
26: end if
```

4.5.1 Exemplos usando grades refinadas

Avaliamos a estratégia de refinamento local executando o algoritmo Refina_Grade (4.5.1) sobre a função *gauss* mostrada na tabela 4.2. Desta forma, encontramos uma base $\mathcal{B}_{min}$ irregular de tendas com profundidade máxima 10, que utilizamos para encontrar uma nova aproximação u^* para a função *gauss*, também aplicando a técnica de mínimos quadrados.

Na figura 4.6 mostramos em curvas de nível o resultado desta aproximação, juntamente com as funções u^* obtidas através das bases $\mathcal{B}_{min}$ regulares de níveis 8 e 10, para fins de comparação.

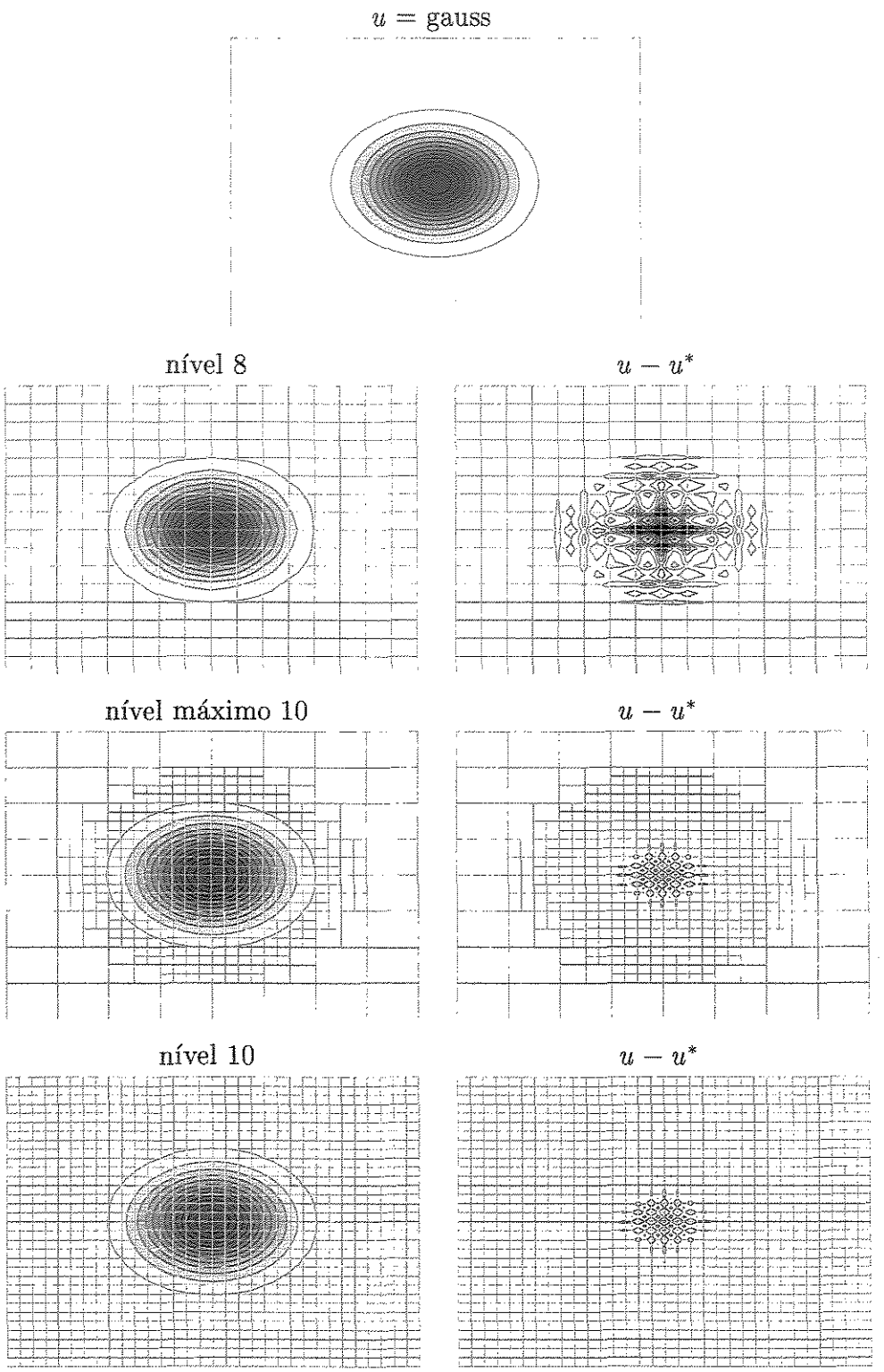


Figura 4.6: Função *gauss* e aproximações nas bases $\mathcal{B}_{min}$ regulares de níveis 8, 10 e em uma base $\mathcal{B}_{min}$ irregular de nível máximo 10, com $\delta_1 = 0,08$ e $\delta_2 = 1,2$. Os resíduos $u - u^*$ possuem $\delta_1 = 0,008$ e $\delta_2 = 0,12$.

As distâncias $\|u - u^*\|$ e a dimensão de cada base são dadas pela tabela 4.4 abaixo:

nível	6	8	máximo 10	10
dimensão	49	225	389	961
$\|u - u^*\|$	0,01386992	0,00297978	0,00069021	0,00068360

Tabela 4.4: Dimensões das bases e distâncias $\|u - u^*\|$ das aproximações.

A tabela 4.4 e a figura 4.6 indicam que com o refinamento local obtivemos uma aproximação da função *gauss* quase tão boa quanto a aproximação realizada através da base regular mais fina. Além disso, a base irregular possui dimensão próxima à dimensão da base regular de nível 8, com bem menos tendas do que a base regular de nível 10. Isto confirma as vantagens do refinamento local neste tipo de aproximação e a eficácia do algoritmo Refina_Grade.

Capítulo 5

Solução de equações diferenciais parciais lineares

Neste capítulo mostramos como resolver equações diferenciais parciais lineares pelo método de Galerkin, descrito num contexto geral.

5.1 O método dos resíduos ponderados

Consideremos uma equação diferencial geral,

$$\mathcal{L}(u)(x) = 0 \qquad x \in \Omega \tag{5.1}$$

onde $\mathcal{L}$ é um operador diferencial (possivelmente não linear), e $u(x)$ é a função que queremos determinar, definida sobre um domínio Ω de dimensão k. Uma vez que em geral não existe uma expressão analítica para $u(x)$, buscamos uma aproximação $u^*(x)$ dentro de um espaço de funções $\mathcal{S}^*$ com dimensão finita n - polinômios, séries trigonométricas, etc. A função $u^*(x)$ é portanto uma combinação linear das funções de uma base qualquer $\varphi = \langle \varphi_0, \ldots, \varphi_{n-1} \rangle$ de $\mathcal{S}^*$:

$$u^*(x) = \sum_{j=0}^{n-1} a_j \varphi_j(x)$$

Desta forma, para obtermos $u^*(x) \simeq u(x)$, basta encontrarmos os coeficientes $a_1, a_2, \ldots, a_n$ mais adequados.

Idealmente, deveríamos procurar os coeficientes que minimizam a diferença $\|u(x) - u^*(x)\|$ entre a solução aproximada e a solução exata. Entretanto, isto

não é praticável em geral. No método dos resíduos ponderados, escolhemos os coeficientes que minimizam o lado esquerdo da equação diferencial (5.1). Embora não seja possível exigir que $\mathcal{L}(u^*)(x)$ seja zero para todo x, podemos exigir que ele satisfaça n equações integrais da forma

$$\int_\Omega \mathcal{L}(u^*)(x) w_i(x) dx = 0 \qquad i = 0, \ldots, n-1$$

onde cada w_i é uma função-peso arbitrária e $dx = dx_{k-1} \ldots dx_0$.

5.2 O método de Galerkin

No método de Galerkin, as funções de ponderação w_i são as próprias funções φ_i da base de aproximação. Portanto, as equações ficam

$$\int_\Omega \mathcal{L}(u^*)(x) \varphi_i(x) dx = 0 \qquad i = 0, \ldots, n-1 \tag{5.2}$$

que podem ser escritas como

$$\langle \mathcal{L}(u^*), \varphi_i \rangle = 0 \qquad i = 0, \ldots, n-1$$

Ou seja, no método de Galerkin procuramos tornar o lado esquerdo de (5.1) ortogonal às funções da base. Substituindo $u^*(x)$ por $\sum_{j=0}^{n-1} a_j \varphi_j(x)$, podemos escrever a equação (5.2) como $r_i(a) = 0$, onde a é o vetor $(a_1, \ldots, a_n)$ e

$$r_i(a) = \int_\Omega \mathcal{L}(\sum_{j=0}^{n-1} a_j \varphi_j)(x) \varphi_i(x) dx \qquad i = 0, \ldots, n-1 \tag{5.3}$$

5.3 Equações lineares

Note que se $\mathcal{L}$ tem a forma $\mathcal{L}(u) = \mathcal{A}(u) + g(x)$, onde $\mathcal{A}$ é um operador diferencial linear, esta expressão pode ser reescrita na forma

$$r_i(a) = \int_\Omega \left(\mathcal{A}(\sum_{j=0}^{n-1} a_j \varphi_j)(x) + g(x) \right) \varphi_i(x) dx = \qquad i = 0, \ldots, n-1 \tag{5.4}$$

$$\sum_{j=0}^{n-1} a_j \int_\Omega \mathcal{A}(\varphi_j)(x) \varphi_i(x) dx + \int_\Omega g(x) \varphi_i(x) dx \qquad i = 0, \ldots, n-1 \tag{5.5}$$

Portanto, as equações $r_i(a) = 0$ formam um sistema também linear de n equações, nas variáveis $a_1, \ldots, a_n$, que pode ser resolvido diretamente. Neste caso, o método de Galerkin geralmente é bastante eficaz [Joh87].

5.3.1 Reduzindo a ordem das derivadas

Quando a equação (5.2) possui alguma integral na forma

$$\int_{\Omega} (\nabla \cdot W)\varphi \, dx \tag{5.6}$$

onde W é uma fórmula vetorial envolvendo derivadas e φ é uma função da base, podemos reduzir a ordem das derivadas utilizando o teorema da divergência [Lei94]. Este teorema afirma que

$$\nabla \cdot (W\varphi) = W \cdot (\nabla\varphi) + (\nabla \cdot W)\varphi \tag{5.7}$$

portanto

$$(\nabla \cdot W)\varphi = \nabla \cdot (W\varphi) - W \cdot (\nabla\varphi)$$

Integrando os dois lados desta expressão, encontramos a equação

$$\int_{\Omega} (\nabla \cdot W)\varphi dx = \int_{\Omega} \nabla \cdot (W\varphi) dx - \int_{\Omega} W \cdot (\nabla\varphi) dx \tag{5.8}$$

Pelo teorema de Green [Lei94], o primeiro termo no lado direito da fórmula (5.8) pode ser reescrito como

$$\int_{\Omega} \nabla \cdot (W\varphi) dx = \int_{\partial\Omega} (W\varphi) \cdot \alpha ds$$

onde $\partial\Omega$ representa a fronteira da região Ω, α é o vetor normal à fronteira num determinado ponto, e ds é um elemento infinitésimo dessa fronteira. Logo, a fórmula (5.8) fica

$$\int_{\Omega} (\nabla \cdot W)\varphi dx = \int_{\partial\Omega} (W\varphi) \cdot \alpha ds - \int_{\Omega} W \cdot (\nabla\varphi) dx \tag{5.9}$$

No nosso caso, Ω é a célula raiz de uma grade diádica e φ é uma tenda, que é zero na fronteira $\partial\Omega$ da mesma. Portanto a expressão (5.6) reduz-se a

$$\int_{\Omega} \varphi(\nabla \cdot W) dx = -\int_{\Omega} (\nabla\varphi) \cdot W dx \tag{5.10}$$

5.3.2 A equação de Helmholtz

Como exemplo consideremos a *equação de Helmholtz*

$$\nabla^2 u(x) + bu(x) = f(u(x), x) \tag{5.11}$$

onde ∇^2 é o operador laplaciano, b é uma constante e f é uma função de u e de x. Note que, quando f não depende da solução u, a equação (5.11) é linear. A equação de Helmholtz é muito usada em geociência, meteorologia, etc. [Gom99, Vve93]

Substituindo u pela aproximação u^*, e aplicando o método de Galerkin (equação (5.2)), obtemos

$$\int_\Omega (\nabla^2 u^* + bu^*)(x)\varphi_i(x)dx = \int_\Omega f(u^*(x), x)\varphi_i(x)dx \qquad i = 0, \ldots, n-1 \tag{5.12}$$

Para simplificar, omitiremos novamente o argumento "(x)" nas integrais que seguem. Escrevendo u^* como a combinação linear $\sum_{j=0}^{n-1} a_j\varphi_j$ e desenvolvendo o lado esquerdo da equação (5.12), obtemos

$$\int_\Omega \nabla^2(\sum_{j=0}^{n-1} a_j\varphi_j)\varphi_i dx + b\int_\Omega (\sum_{j=0}^{n-1} a_j\varphi_j)\varphi_i dx =$$

$$\sum_{j=0}^{n-1} a_j \int_\Omega (\nabla^2\varphi_j)\varphi_i dx + b\sum_{j=0}^{n-1} a_j \int_\Omega \varphi_j\varphi_i dx \tag{5.13}$$

Reduzindo a ordem das derivadas, conforme a seção 5.3.1, decompomos o operador laplaciano na expresão (5.13) em termos de gradientes, na forma

$$\sum_{j=0}^{n-1} a_j \int_\Omega (\nabla\varphi_j)\cdot(\nabla\varphi_i)dx + b\sum_{j=0}^{n-1} a_j \int_\Omega \varphi_j\varphi_i dx =$$

$$\sum_{j=0}^{n-1} a_j \int_\Omega (\nabla\varphi_j)\cdot(\nabla\varphi_i) + b(\varphi_j\varphi_i)dx$$

de maneira que obtemos o sistema

$$\sum_{j=0}^{n-1} a_j \int_\Omega (\nabla\varphi_j)\cdot(\nabla\varphi_i) + b(\varphi_j\varphi_i)dx = \int_\Omega f\varphi_i dx \qquad i = 0, \ldots, n-1$$

Em forma matricial, o sistema é $Ea = g$, onde a é o vetor (coluna) dos coeficientes de u^*, E é a matriz $n \times n$ tal que

$$E_{ij} = \int_\Omega (\nabla\varphi_j) \cdot (\nabla\varphi_i) + b(\varphi_j\varphi_i)dx$$

e g é o vetor (coluna) tal que $g_i = \int_\Omega f\varphi_i dx$.

5.3.3 Produto escalar de gradientes de funções tenda

O produto escalar $\langle \nabla\tau', \nabla\tau'' \rangle$ dos gradientes de duas tendas, assim como o produto escalar entre elas (seção 4.2), também pode ser decomposto na soma de integrais sobre as células da tenda de maior profundidade contidas no domínio da tenda de menor profundidade. Ou seja,

$$\langle \nabla\tau', \nabla\tau'' \rangle = \int_\Omega \sum_{i=0}^{k-1} \left(\frac{\partial\tau'}{\partial x_i}\right)\left(\frac{\partial\tau''}{\partial x_i}\right) dx = \int_{C'_j} \sum_{i=0}^{k-1} \left(\frac{\partial\tau'}{\partial x_i}\right)\left(\frac{\partial\tau''}{\partial x_i}\right) dx \qquad (5.14)$$

onde C'_j é uma célula do suporte D' da tenda τ', o qual supomos ser menor do que o suporte D'' da tenda τ'', e $C'_j \in (D' \cap D'')$.

Desta forma, usamos a mesma técnica da seção (4.2) para calcularmos $\langle \nabla\tau', \nabla\tau'' \rangle$. A partir da definição (3.1) de uma tenda, podemos escrever os gradientes $\nabla\tau'$ e $\nabla\tau''$ como dois vetores de k elementos tais que

$$(\nabla\tau')_i = (2^{i/k}Q'_i)\,\Gamma(2^{i/k}Q'_i x_i - \delta'_i) \prod_{j=0, j\neq i}^{k-1} \Lambda(2^{j/k}Q'_j x_j - \delta'_j) \qquad (5.15)$$

e

$$(\nabla\tau'')_i = (2^{i/k}Q'_i\alpha_i)\,\Gamma(\alpha_i(2^{i/k}Q'_i x_i - \delta'_i) + \beta_i) \prod_{j=0, j\neq i}^{k-1} \Lambda(\alpha_j(2^{j/k}Q'_j x_j - \delta'_j) + \beta_j) \qquad (5.16)$$

onde Γ é a derivada da função Λ (figura 5.1), definida por

$$\Gamma(x) = \begin{cases} -1 & \text{se } 0 \leq x \leq 1 \\ 0 & \text{se } |x| > 1 \\ +1 & \text{se } -1 \leq x < 0 \end{cases} \qquad (5.17)$$

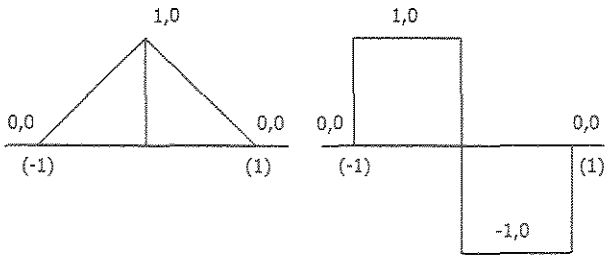


Figura 5.1: Funções Λ (esquerda) e Γ (direita).

O produto escalar $\langle \nabla\tau', \nabla\tau'' \rangle$, restrito a C', é dado por

$$\int_{x_0^o}^{x_0^f} \cdots \int_{x_{k-1}^o}^{x_{k-1}^f} \sum_{i=0}^{k-1} \mu_i \prod_{j=0, j\neq i}^{k-1} \Lambda(2^{j/k} Q'_j x_j - \delta'_j) \Lambda(\alpha_i (2^{j/k} Q'_j x_j - \delta'_j) + \beta_i) dx_{k-1} \ldots dx_0 \tag{5.18}$$

tal que

$$\mu_i = \Gamma(2^{i/k} Q'_i x_i - \delta_i)\, (2^{i/k} Q'_i)\, \Gamma(\alpha_i (2^{i/k} Q'_i x_i - \delta_i) + \beta_i)\, (2^{i/k} Q'_i \alpha_i)$$

onde os valores de x_i^o, x_i^f, α_i e β_i são os mesmos da seção 4.2, usados na equação (4.8). Desenvolvendo a equação (5.18), obtemos

$$\sum_{i=0}^{k-1} \int_{x_0^o}^{x_0^f} \cdots \int_{x_{k-1}^o}^{x_{k-1}^f} \mu_i \prod_{j=0, j\neq i}^{k-1} \Lambda(2^{j/k} Q'_j x_j - \delta'_j) \Lambda(\alpha_i (2^{j/k} Q'_j x_j - \delta'_j) + \beta_i) dx_{k-1} \ldots dx_0 \tag{5.19}$$

Separando as integrais, e observando que μ_i é constante dentro da célula C',

$$\sum_{i=0}^{k-1} \mu_i \left(\int_{x_i^o}^{x_i^f} 1 dx_i \right) \prod_{j=0, j\neq i}^{k-1} \left(\int_{x_j^o}^{x_j^f} \Lambda(2^{j/k} Q'_j x_j - \delta'_j) \Lambda(\alpha_i (2^{j/k} Q'_j x_j - \delta'_j) + \beta_i) dx_j \right) \tag{5.20}$$

Considerando que $\int_{x_i^o}^{x_i^f} 1 dx_i = x_i^f - x_i^o = (2^{i/k} Q'_i)^{-1}$, e realizando as mudanças de variáveis (4.10), obtemos

$$\sum_{i=0}^{k-1} \mu_i (2^{i/k} Q'_i)^{-1} \prod_{j=0, j\neq i}^{k-1} (2^{j/k} Q'_j)^{-1} \int_{u_j^o}^{u_j^f} \Lambda(u_j) \Lambda(\alpha_i u_j + \beta_i) du_j = \tag{5.21}$$

$$\frac{1}{\prod_{j=0}^{k-1} (2^{j/k} Q'_j)} \sum_{i=0}^{k-1} \mu_i \prod_{j=0, j\neq i}^{k-1} \int_{u_j^o}^{u_j^f} \Lambda(u_j) \Lambda(\alpha_i u_j + \beta_i) du_j \tag{5.22}$$

As integrais $\int_{u_j^o}^{u_j^f} \Lambda(u_j) \Lambda(\alpha_i u_j + \beta_i) du_j$ podem ser calculadas através das fórmulas (4.11) a (4.14).

5.4 Exemplo

Para testar numericamente esta teoria, resolvemos especificamente a seguinte equação de Helmholtz:

$$\nabla^2 u + 0,5u = -9\pi^2 + 0,5(\sin(\pi x)\sin(2\sqrt{2}\pi y)) \tag{5.23}$$

cuja solução exata é a função

$$u = \sin(\pi x)\sin(2\sqrt{2}\pi y) \tag{5.24}$$

Usamos 4 bases $\mathcal{B}_{min}$ de aproximação, extraídas das grades diádicas uniformes de profundidades 6, 8, 10, e de uma grade irregular de profundidade 10. Construímos a grade irregular executando o algoritmo 4.5.1 (Refina_Grade) sobre a função (5.24), representada em curvas de nível na figura 5.2. As soluções aproximadas e o erro $\|u - u^*\|$ de cada teste são mostrados na figura 5.3.

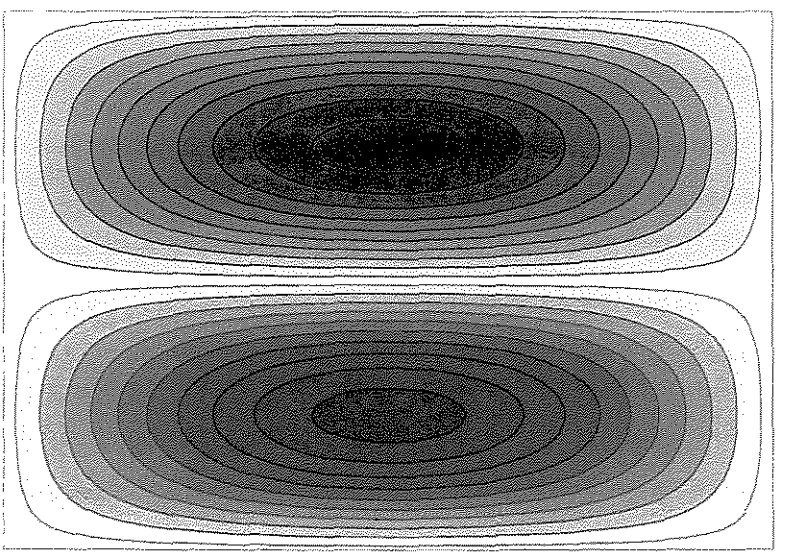


Figura 5.2: Solução exata da equação (5.23) de Helmholtz, com $\delta_1 = 0,1$ e $\delta_2 = 1,1$.

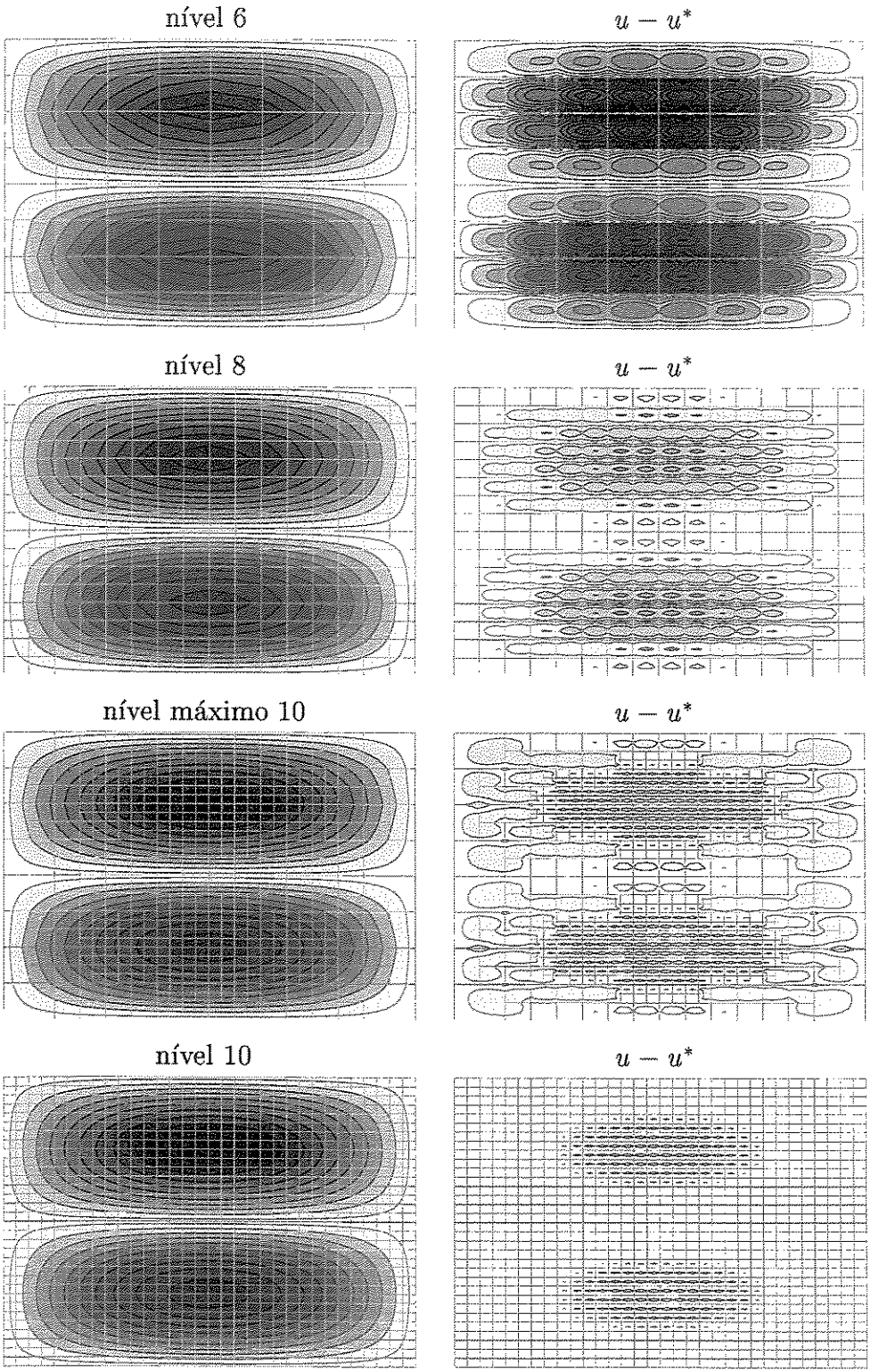


Figura 5.3: Solução da equação de Helmholtz usando bases minimais definidas sobre grades regulares de níveis 6, 8, 10 e uma grade irregular de profundidade máxima 10, com $\delta_1 = 0,1$ e $\delta_2 = 1,1$. Os resíduos $u - u^*$ possuem $\delta_1 = 0,006$ e $\delta_2 = 0,07$.

Na tabela 5.1 a seguir, estão as distâncias $\|u - u^*\|$, as dimensões das bases utilizadas e os valores dos diâmetros λ das menores células de cada grade.

nível	6	8	máximo 10	10
λ	0,1531	0,0765	0,0383	0,0383
dimensão	49	225	439	961
$\|u - u^*\|$	0,02192191	0,00550957	0,00331064	0,00137926

Tabela 5.1: Dimensões das bases e distâncias $\|u - u^*\|$ das soluções aproximadas de uma equação de Helmholtz.

Assim como nos exemplos anteriores, os erros de aproximação são quase proporcionais a λ^2 nas aproximações pelas bases regulares de níveis 6, 8 e 10 (com proporções de aproximadamente 16 : 4 : 1). Utilizando a base adaptativa, reduzimos a dimensão do espaço de aproximação para menos da metade da dimensão da base regular de nível 10 (dimensão $439 < 961/2$), diminuindo portanto o custo computacional. O erro relativo a esta base adaptativa, contudo, foi maior do que o dobro do erro relativo à mesma base regular para o caso estudado.

Capítulo 6

Solução de equações diferenciais lineares e dependentes do tempo

Estenderemos agora a discussão do capítulo anterior para abordarmos um caso particular de equação diferencial parcial, com a forma

$$\mathcal{L}(u)(x,t) = 0 \qquad x \in \Omega \tag{6.1}$$

onde $\mathcal{L}$ é novamente um operador diferencial, x é um ponto do domínio Ω, com dimensão k, e $u(x,t)$ é a função que queremos encontrar, variável no espaço (x) e no tempo (t). Poderíamos simplesmente usar o método de Galerkin mostrado na seção 5.2 e resolver a equação (6.1) tratando o tempo t e as outras x_i dimensões da mesma forma. Entretanto, para utilizarmos tendas como base de aproximação, teríamos que construir uma grade diádica G' com dimensão $k+1$, na qual os refinamentos no tempo estariam atrelados a refinamentos no espaço e vice-versa, o que pode ser altamente indesejável considerando que estas grandezas não possuem uma relação direta.

Nestas situações, é comumente utilizada uma solução alternativa em que o método de Galerkin é aplicado apenas nas dimensões espaciais (x), e o método de *integração de Euler* [PFTV86] é usado na direção temporal. Mais precisamente, para cada instante t, usamos o método de Galerkin para calcular a derivada temporal $\frac{\partial u}{\partial t}$. Calculamos então a função u em um instante posterior $t + \Delta t$ como

$$u^{(t+\Delta t)} \leftarrow u^{(t)} + \left(\frac{\partial u}{\partial t}\right)^{(t)} \times \Delta t \tag{6.2}$$

Na verdade, trabalhamos com uma aproximação u^* de u, $u^* = \sum_{i=0}^{n-1} a_i \varphi_i$ onde as

funções φ_i não variam com o tempo t. Portanto precisamos calcular coeficientes $b_0, \ldots, b_{n-1}$ tais que $\frac{\partial u}{\partial t} = \sum_{i=0}^{n-1} b_i \varphi_i$, e então fazer

$$a_i^{(t+\Delta t)} \leftarrow a_i^{(t)} + b_i^{(t)} \times \Delta t \tag{6.3}$$

6.1 A difusão do calor

Para avaliar os espaços de *splines* diádicos $\mathcal{E}^1[G]$ neste tipo de aplicação, realizamos testes com a equação de difusão do calor em meio isotrópico. O fenômeno da transferência de calor é amplamente citado no estudo de equações diferenciais parciais porque serve de modelo para outros fenômenos de difusão [Men77, Twi84, Vic81]. A difusão do calor em meio isotrópico é descrita pela seguinte equação:

$$\frac{\partial T(x,t)}{\partial t} = K(x)\nabla^2 T(x,t) \tag{6.4}$$

onde $T(x,t)$ é a *temperatura* em um ponto $x = (x_0, \ldots, x_{k-1})$ de um domínio Ω k-dimensional, no instante t e $K(x)$ é a *função de condutividade térmica* em x. Esta expressão indica que quando há uma distribuição de temperatura irregular em um corpo ou superfície, então ocorre transferência de calor dos pontos de maior temperatura para os pontos de menor temperatura, até que o corpo alcance temperatura constante.

Esta equação é linear na solução $T(x,t)$. Se o meio for homogêneo – isto é, a função $K(x)$ é constante em Ω – e a distribuição do calor no instante $t = 0$ for um impulso de Dirac [Bra78] centrado no ponto x^o, então a solução analítica [Sew88] da equação (6.4) é

$$T(x) = \frac{e^h}{4\pi K t}, \qquad h = -\frac{\sum (x_i - x_i^o)^2}{4Kt} \tag{6.5}$$

Esta função é uma gaussiana simétrica com centro em $(x_0^o, \ldots, x_{k-1}^o)$.

Na solução numérica, começamos aplicando a condição de Galerkin (seção 5.2) à equação (6.4) apenas no domínio espacial Ω, obtendo

$$\int_\Omega \left(\frac{\partial T}{\partial t} - K\nabla^2 T \right) \varphi_i dx = 0 \qquad i = 0, \ldots, n-1$$

ou seja

$$\int_\Omega \frac{\partial T}{\partial t} \varphi_i dx = K \int_\Omega (\nabla^2 T)\varphi_i dx \qquad i = 0, \ldots, n-1 \tag{6.6}$$

onde $dx = dx_{k-1} \ldots dx_0$. Substituindo T por $\sum_{j=0}^{n-1} a_j \varphi_j$ e $\frac{\partial T}{\partial t}$ por $\sum_{j=0}^{n-1} b_j \varphi_j$, reescrevemos a equação (6.6) para cada i como

$$\int_\Omega (\sum_{j=0}^{n-1} b_j \varphi_j) \varphi_i dx = K \int_\Omega (\nabla^2 \sum_{j=0}^{n-1} a_j \varphi_j) \varphi_i dx =$$

$$\sum_{j=0}^{n-1} b_j \int_\Omega \varphi_j \varphi_i dx = K \sum_{j=0}^{n-1} a_j \int_\Omega (\nabla^2 \varphi_j) \varphi_i dx \tag{6.7}$$

Reduzindo a ordem das derivadas no lado direito da equação (6.7) através dos teoremas de Green e da divergência (seção 5.3.1), obtemos

$$\sum_{j=0}^{n-1} b_j \int_\Omega \varphi_j \varphi_i dx = -K \sum_{j=0}^{n-1} a_j \int_\Omega (\nabla \varphi_j) \cdot (\nabla \varphi_i) dx \tag{6.8}$$

ou seja, a equação (6.6) corresponde a um sistema com representação matricial $Eb = c$, no qual E é a matriz $n \times n$ com $E_{ij} = \langle \varphi_i, \varphi_j \rangle$, e c é o vetor (coluna) tal que $c_i = K \sum_{j=0}^{n-1} a_j \langle \nabla \varphi_i, \nabla \varphi_j \rangle$. Supondo que os coeficientes a_j em um dado instante sejam conhecidos, resolvendo o sistema encontramos os coeficientes b_j neste mesmo instante. Com estes valores, calculamos os coeficientes $a^{(t+\Delta t)}$ em função dos coeficientes $a^{(t)}$ através da fórmula (6.3) da integração de Euler.

6.2 Exemplo de simulação

Nos testes numéricos, simulamos o processo descrito pela equação (6.4) no intervalo de $t^o = 0,0004s$ a $t^f = 0,006s$, com $\Delta t = 0.000007s$, usando como vetor inicial a os coeficientes da aproximação por mínimos quadrados na base $\varphi = \mathcal{B}_{min}$ (seção 4.1) da solução exata (6.5), para o instante inicial ($t = t^o$). Comparamos o resultado final da integração $T^{*f}(x) = T^*(x, t^f)$ com a aproximação $T^f(x)$ da solução exata $T(x, t^f)$ (6.5) no instante final ($t = t^f$). Desta forma, o erro $\left\| T^f - T^{*f} \right\|$ reflete apenas os erros de integração no tempo, e não o erro (inevitável) da aproximação da solução T^f no espaço $\mathcal{E}^1[G]$.

Empregamos as mesmas bases regulares usadas nos experimentos anteriores, com tendas de profundidades 6, 8 e 10. Utilizamos também uma base irregular, contendo tendas de profundidade máxima 10, sendo que todas as 4 bases citadas são minimais. A base irregular foi obtida através do algoritmo 4.5.1, Refina_Grade, aplicado à solução final $T(x, t^f)$.

Nas figuras 6.1 a 6.4 exibimos 10 estágios de simulação para cada base, desde a temperatura no instante inicial ($T^*(x, t^o)$) até o resultado final de integração (T^{*f}). Na última linha de cada sequência mostramos também a solução (T^f) e o erro ($T^f - T^{*f}$).

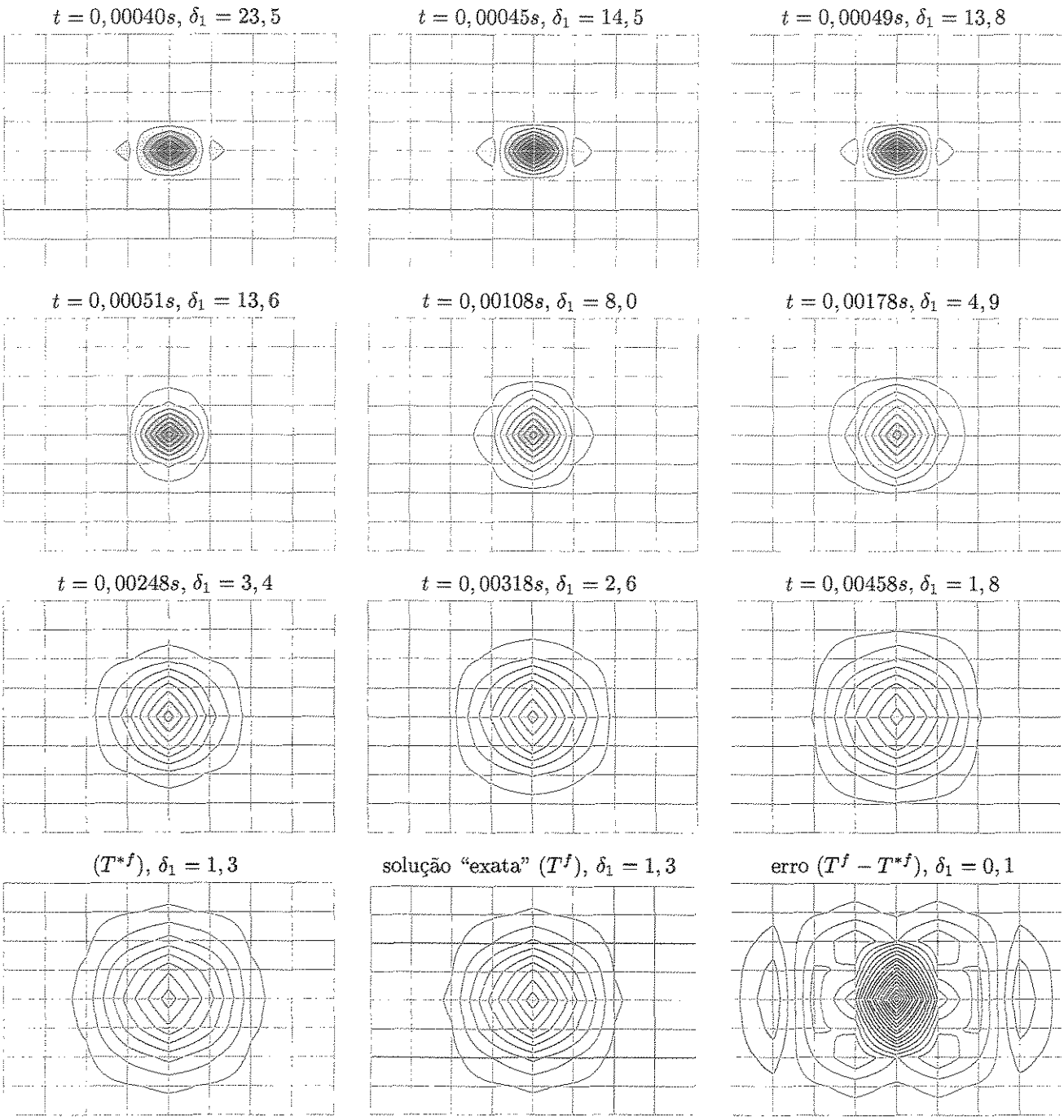


Figura 6.1: Simulação da difusão do calor com uma base regular de profundidade 6. A última linha mostra o resultado final da integração (T^{*f}), a aproximação por mínimos quadrados da solução exata correspondente (T^f), e a diferença entre estas duas. Todas as imagens têm parâmetro $\delta_2 = 188,0$.

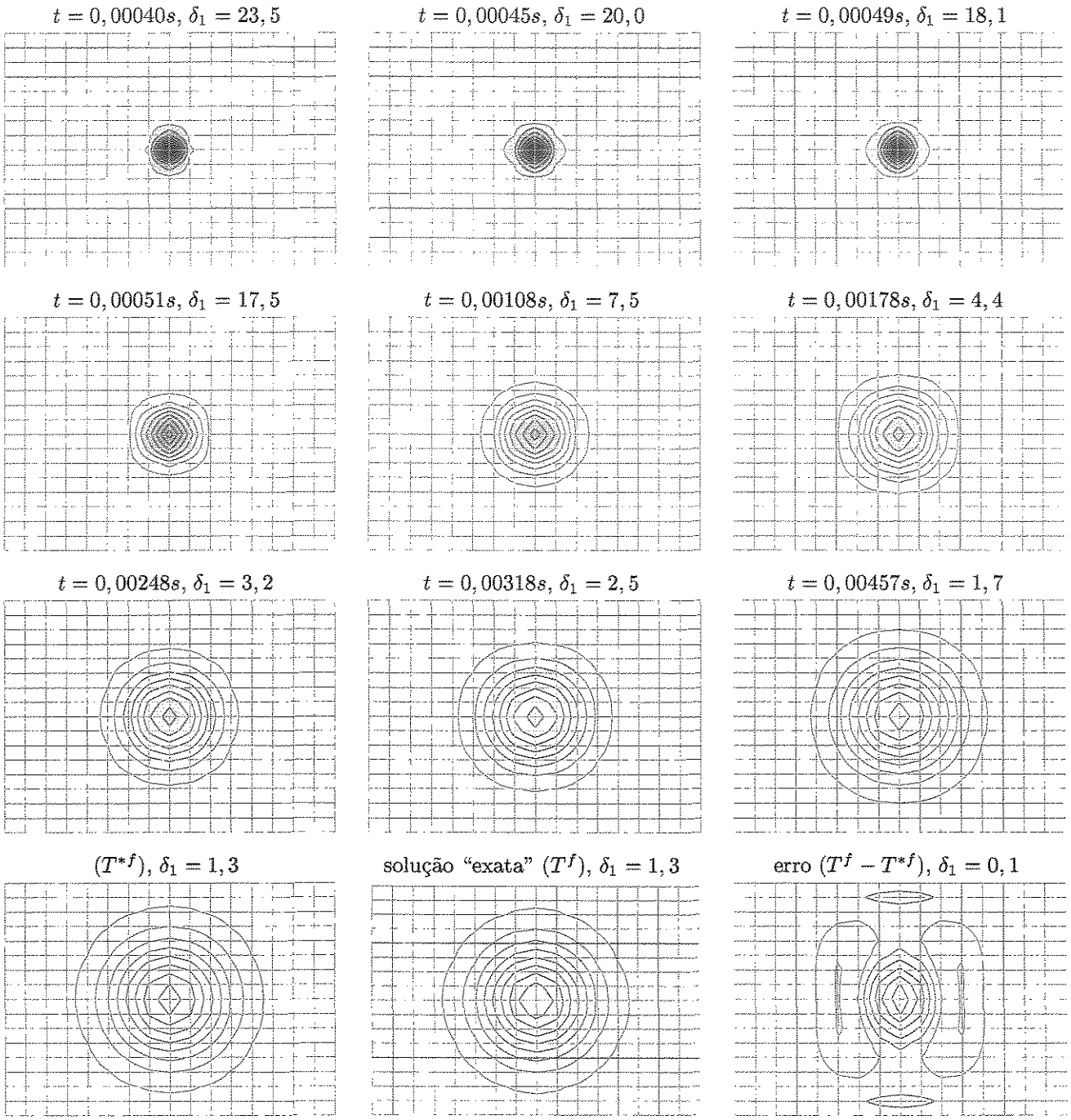


Figura 6.2: Simulação da difusão do calor com uma base regular de profundidade 8. A última linha mostra o resultado final da integração (T^{*f}), a aproximação por mínimos quadrados da solução exata correspondente (T^f), e a diferença entre estas duas. Todas as imagens têm parâmetro $\delta_2 = 188,0$.

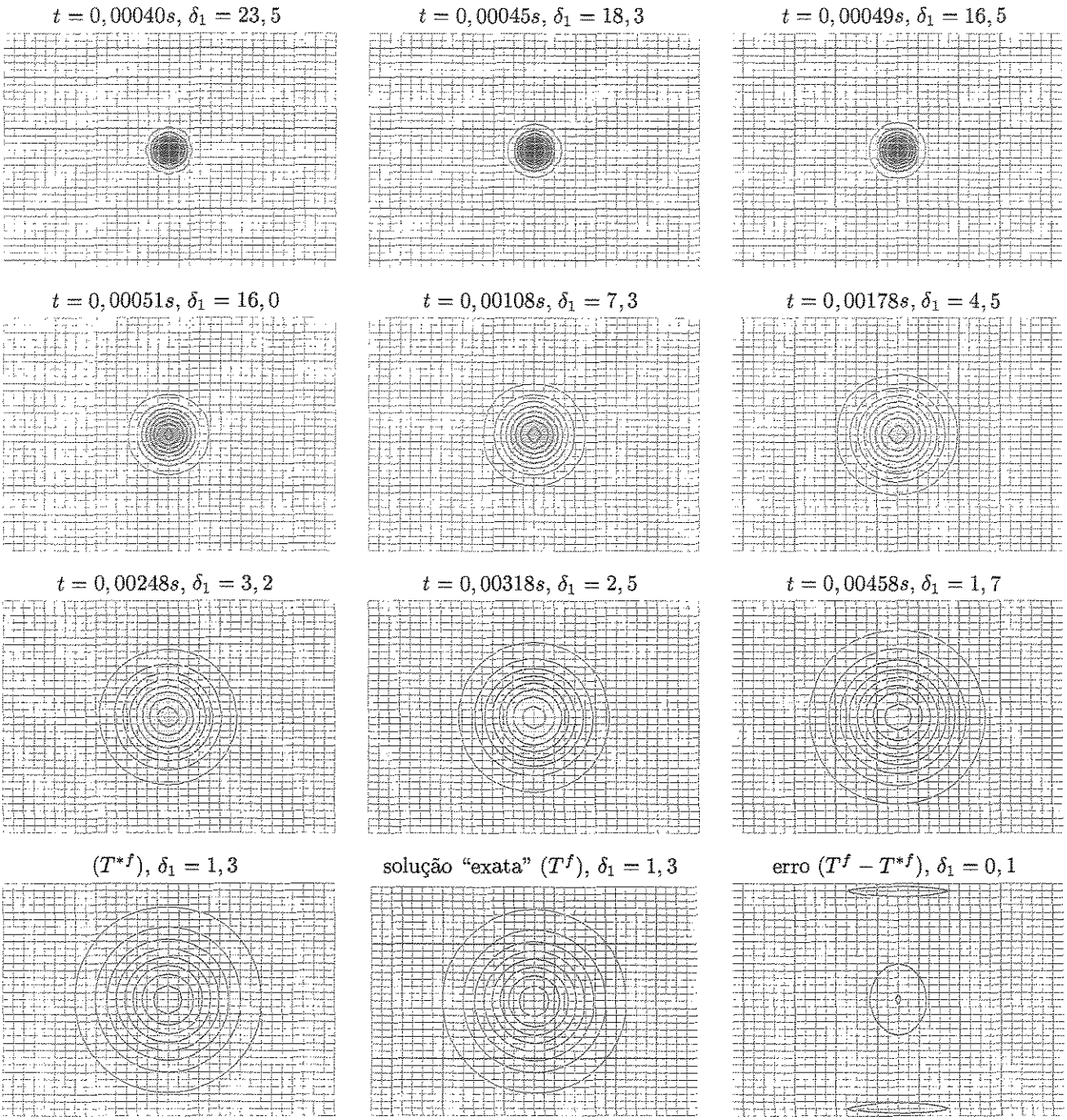


Figura 6.3: Simulação da difusão do calor com uma base regular de profundidade 10. A última linha mostra o resultado final da integração (T^{*f}), a aproximação por mínimos quadrados da solução exata correspondente (T^f), e a diferença entre estas duas. Todas as imagens têm parâmetro $\delta_2 = 188,0$.

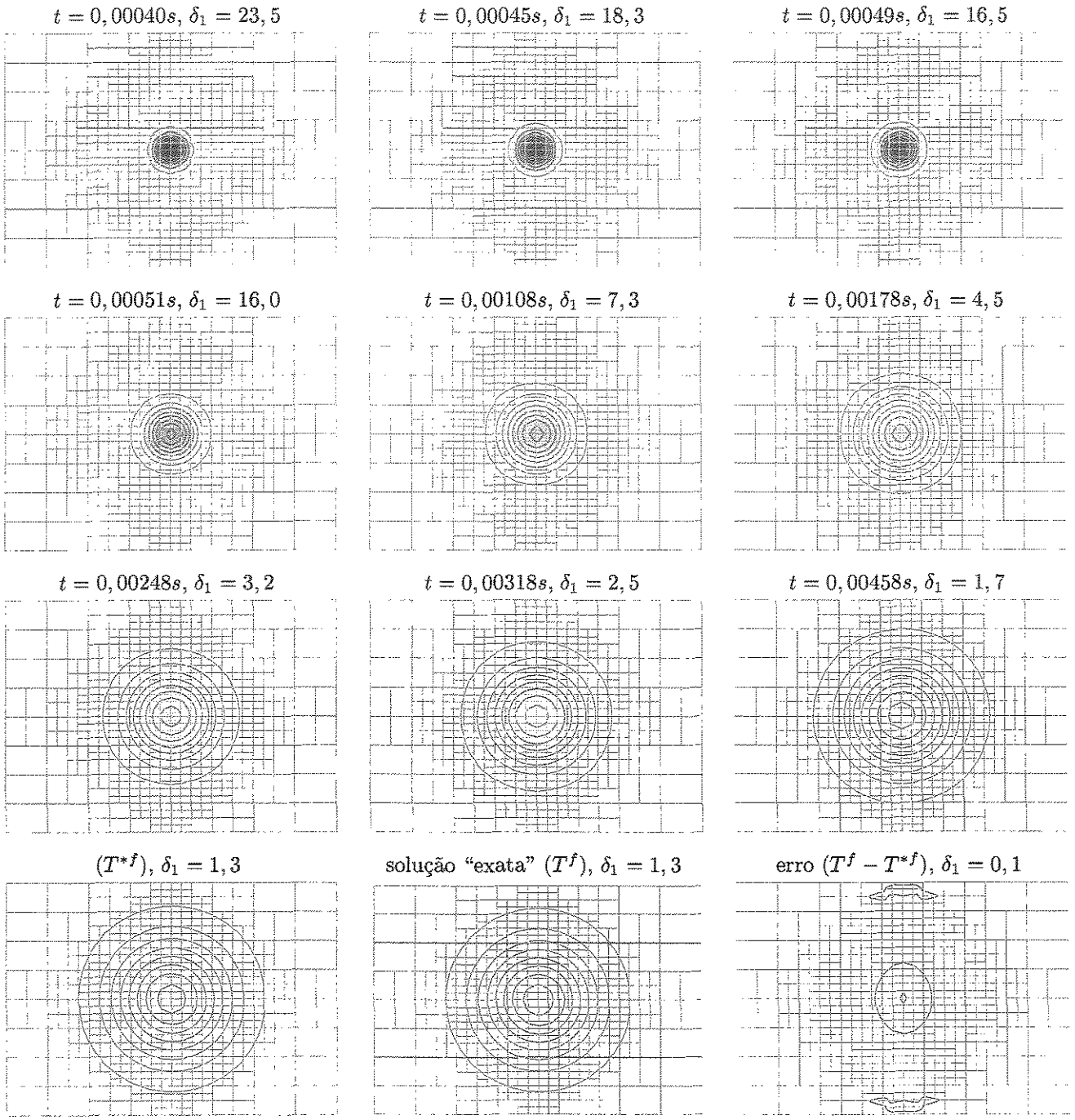


Figura 6.4: Simulação da difusão do calor com uma base irregular de profundidade máxima 10. A última linha mostra o resultado final da integração (T^{*f}), a aproximação por mínimos quadrados da solução exata correspondente (T^f), e a diferença entre estas duas. Todas as imagens têm parâmetro $\delta_2 = 188,0$.

Na tabela 6.1 a seguir mostramos as distâncias $\|T^f - T^{*f}\|$, as dimensões das bases e o diâmetro λ das menores células de cada grade.

nível	6	8	máximo 10	10
λ	0,1531	0,0765	0,0383	0,0383
dimensão	49	225	437	961
$\|u - u^*\|$	0,17572679	0,07047778	0,02286419	0,02344591

Tabela 6.1: Dimensões das bases e distâncias $\|T^f - T^{*f}\|$ entre os resultados finais de quatro simulações de um processo de difusão.

De acordo com as distâncias $\|u - u^*\|$ (tabela 6.1), o ganho de precisão devido ao refinamento das grades na integração no tempo foi um pouco menor do que o ganho de precisão obtido com este refinamento nos exemplos de integração espacial (seção 5.4). Entretanto, na simulação realizada com a grade irregular de profundidade máxima 10, obtivemos um resultado tão preciso quanto o resultado obtido por intermédio da grade regular mais fina, de nível 10, com uma quantidade significativamente menor de elementos na base. Neste sentido a integração no tempo com refinamento local mostrou-se bastante vantajosa.

Capítulo 7

As equações de escoamento

A partir deste capítulo nos voltamos especificamente para o problema do escoamento bifásico óleo/água em meios porosos. Começamos revisando o sistema de equações diferenciais não linear e dependente do tempo que descreve este processo físico [AS79, Cri77], desde uma abordagem geral até nosso caso particular, que depois será discretizado e modelado por funções tenda.

7.1 Modelagem matemática

O escoamento de petróleo em meios porosos pode ser descrito por um conjunto de equações diferenciais parciais não lineares. As equações refletem as leis de conservação de massa, viscosidade e pressão. Para obter as equações consideradas, tomamos como base um elemento infinitesimal de volume com dimensão dx, dy e dz, contendo rocha e fluidos, como ilustrado na figura 7.1.

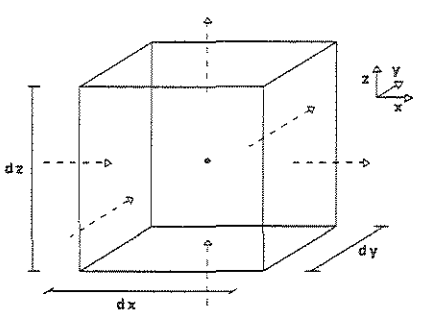


Figura 7.1: Elemento infinitesimal $dx \times dy \times dz$

A lei da conservação de massa diz que, num determinado intervalo de tempo dt,

$$m_e - m_s + m_p = m_a \tag{7.1}$$

onde:

$m_e - m_s$ é a diferença entre a massa que entra e a massa que sai através de todas as faces do elemento;

m_p é a massa que entra ou sai do elemento através dos poços produtores ou injetores;

m_a é o incremento na massa de fluido presente no volume.

Definimos a velocidade de escoamento v_x, v_y, v_z da seguinte forma: v_x é o volume de fluido dV que atravessa um elemento infinitesimal de área dA perpendicular ao eixo x, num intervalo de tempo dt, dividido por estes dois,

$$v_x = \frac{dV_x}{dtdA_x}$$

e igualmente para os demais eixos

$$v_y = \frac{dV_y}{dtdA_y}, \quad \text{e} \quad v_z = \frac{dV_z}{dtdA_z}$$

O vetor $v = (v_x, v_y, v_z)$ é a *velocidade do escoamento.*

Portanto, se considerarmos apenas as faces perpendiculares ao eixo x, a parcela correspondente da diferença $m_e - m_s$ é dada por

$$-\frac{\partial}{\partial x}(\rho v_x)dydzdxdt$$

onde ρ é a *densidade.* Se considerarmos o fluxo nas direções x, y e z, temos

$$m_e - m_s = -\frac{\partial}{\partial x}(\rho v_x)dydzdxdt - \frac{\partial}{\partial y}(\rho v_y)dzdxdydt - \frac{\partial}{\partial z}(\rho v_z)dxdydzdt$$

Em forma vetorial, usando o operador $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$ podemos escrever

$$m_e - m_s = -\nabla.(\rho v)dxdydzdt$$

Por outro lado, a massa dos poços pode ser escrita como

$$m_p = \rho QBdt$$

onde Q é a *vazão* (volume por unidade de tempo) medida na superfície, e B é o *fator volume de formação*, que representa a relação entre o volume do fluido no reservatório e seu volume em condições padrões. Finalmente, a variação na massa acumulada é dada por

$$m_a = \frac{\partial}{\partial t}(\phi\rho)dxdydzdt$$

onde ϕ é a *porosidade* da rocha. A porosidade ϕ indica a fração do volume que não está ocupada pela rocha, mas pelo fluido.

Assim, a equação (7.1) assume a forma

$$-\nabla.(\rho v)dxdydzdt = \frac{\partial}{\partial t}(\phi\rho)dxdydzdt - \rho QBdt \tag{7.2}$$

7.1.1 A lei de Darcy

A velocidade v_x do escoamento, por sua vez, é dada pela fórmula empírica conhecida como *Lei de Darcy*. Para meios isotrópicos,

$$v_x = -\frac{K}{\mu}\left(\frac{\partial p}{\partial x} - \rho g_x\right) \quad v_y = -\frac{K}{\mu}\left(\frac{\partial p}{\partial y} - \rho g_y\right) \quad v_z = -\frac{K}{\mu}\left(\frac{\partial p}{\partial z} - \rho g_z\right)$$

onde K é uma propriedade do meio chamada *permeabilidade*, μ é a *viscosidade* do fluido, e g_x é a componente do vetor gravidade na direção do eixo x (que não é necessariamente zero, porque o eixo z não é necessariamente vertical). O fator $\frac{\partial p}{\partial x} - \rho g_x$ é denominado *diferencial do potencial*, e é a força que obriga as partículas do fluido a se moverem. Em forma vetorial

$$v = -\frac{K}{\mu}(\nabla p - \rho g) \tag{7.3}$$

No caso de meios não isotrópicos, o fator K deve ser substituído por um tensor, representável por uma matriz

$$K = \begin{bmatrix} k_x & k_{xy} & k_{xz} \\ k_{yx} & k_y & k_{yz} \\ k_{zx} & k_{zy} & k_z \end{bmatrix}$$

Frequentemente as direções principais do tensor coincidem com os eixos de coordenadas. Nestes casos K tem a forma

$$K = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix}$$

Chamamos K de *tensor de permeabilidade absoluta*.

7.2 Escoamento bifásico óleo/água

7.2.1 Saturações relativas

As leis de escoamento exigem algumas adaptações quando descrevem o fluxo de dois ou mais fluidos imiscíveis, chamados *fases*. Nesta situação, utilizamos o conceito de *saturação relativa*, no qual o volume de escoamento é compartilhado entre estes fluidos. As saturações são expressas como frações do volume de poros, ou seja, do volume total menos o volume ocupado pelas rochas. Consequentemente, não podem assumir valores negativos nem superiores a 1, e sua soma é sempre 1. Considerando duas fases apenas, digamos óleo (o) e água (a), temos portanto a equação

$$S_o + S_a = 1 \tag{7.4}$$

onde S_o é a saturação de óleo e S_a é a saturação de água.

Observa-se que o escoamento de uma fase é dificultado pelas outras, mais do que seria causado apenas pela redução do volume de poros. Ou seja, cada fase atrapalha o escoamento das outras de maneira que o fluxo multifásico total é sempre mais lento do que o mesmo fluxo com apenas uma de suas fases. Portanto nas equações a permeabilidade K deve ser multiplicada por um fator k_l, chamado *permeabilidade relativa*, que expressa essa redução. O valor de k_l é próprio de cada fase. A soma das permeabilidades relativas é sempre menor ou igual a 1. Onde uma fase está sozinha ($S_l = 1$) a permeabilidade relativa k_l será 1, mas cada k_l pode ser maior que o S_l correspondente.

7.2.2 Pressões capilares

As pressões de cada fase também possuem diferenças, porque normalmente algumas delas aderem às rochas mais facilmente e exercem uma força de compressão sobre as demais, que precisam compensá-la para manterem o equilíbrio. Portanto em vez de uma única pressão p, devemos considerar uma pressão p_l diferente para cada fase l.

Esta pressão adicional é chamada *pressão capilar*. Quando o meio possui apenas duas fases, a fase m que adere mais facilmente à rocha é dita *molhante* (ou *molhada*) e a outra fase n é dita *não molhante*. Nesse caso a pressão de capilaridade é definida por

$$P_{cnm}(S_n, S_m) = p_n - p_m$$

onde P_{cnm} é uma função da saturação destas fases, S_n e S_m. Em um sistema contendo apenas água e óleo, normalmente a água é a fase molhante e o óleo é a fase não molhante, e portanto escrevemos a pressão de capilaridade P_{coa} do óleo sobre a água como

$$P_{coa}(S_a, S_o) = p_o - p_a \tag{7.5}$$

7.2.3 A equação geral

Levando em conta estas características, podemos re-escrever a relação (7.2) para uma fase l de um fluxo multifásico não miscível como

$$\begin{aligned} &\frac{\partial}{\partial x}\left[\rho_l K \frac{k_l}{\mu_l B_l}\left(\frac{\partial p_l}{\partial x} - \rho_l g_x\right)\right] dydzdxdt + \\ &\frac{\partial}{\partial y}\left[\rho_l K \frac{k_l}{\mu_l B_l}\left(\frac{\partial p_l}{\partial y} - \rho_l g_y\right)\right] dzdxdydt + \\ &\frac{\partial}{\partial z}\left[\rho_l K \frac{k_l}{\mu_l B_l}\left(\frac{\partial p_l}{\partial z} - \rho_l g_z\right)\right] dxdydzdt = \\ &\frac{\partial}{\partial t}\left(\rho_l \frac{S_l \phi}{B_l}\right) dxdydzdt - \rho_l Q_l dt \end{aligned} \tag{7.6}$$

Escrevendo-a na forma de divergente,

$$\nabla.(\rho_l K \frac{k_l}{\mu_l B_l}(\nabla p_l - \rho_l g)) - \frac{\partial}{\partial t}\left(\rho_l \frac{S_l \phi}{B_l}\right) + \rho_l q_l = 0 \tag{7.7}$$

onde q_l é a *taxa de injeção* (volumétrica), que representa o volume de fluido introduzido num dado volume, proveniente de fontes externas ao sistema (no caso, poços de injeção e extração), por unidade de tempo. Quanto q_l assume valores negativos, determina uma taxa de extração. A relação entre a vazão Q_l e a taxa de injeção q_l é expressa por

$$q_l = \frac{Q_l}{dxdydz}$$

O termo k_l depende da saturação da fase l e os termos μ_l, B_l e ρ_l dependem da pressão e temperatura da fase l, logo não podem ser retirados das derivadas da equação (7.7). Além disso, os termos K e ϕ dependem da posição no reservatório. Estas dependências constituem *propriedades de estado* dos fluidos, da rocha ou propriedades rocha-fluido, e são responsáveis pela não linearidade acentuada do sistema final.

Capítulo 8

Discretização das equações

Este capítulo mostra a discretização matemática das equações de escoamento visando a aplicação do método de Galerkin com as bases de funções tenda. Além disso, explica em detalhes a integração no tempo e no espaço destas equações, utilizando tanto grades diádicas estáticas quanto dinâmicas.

8.1 Discretização das equações de escoamento

Neste trabalho utilizamos o modelo físico conhecido na indústria como *Black-Oil*, restrito a duas fases apenas – água ($l = 'a'$) e óleo ($l = 'o'$), com pressão de capilaridade nula ($p_a = p_o$). Neste modelo, as duas fases líquidas são consideradas imiscíveis. Uma vez que as variações de pressão de ponto a ponto são pequenas comparadas com a pressão média no reservatório, é conveniente trabalhar apenas com a diferença $p(x, y, z, t) = p_o(x, y, z, t) - \bar{p} = p_a(x, y, z, t) - \bar{p}$, onde $\bar{p}$ é uma *pressão típica* constante no espaço e no tempo, próxima da pressão média. Podemos portanto resolver a equação (7.7) encontrando as funções

$$p(x, y, z, t) \qquad e \qquad S_l(x, y, z, t)$$

que a satisfazem, dadas as demais funções e parâmetros (ρ, k, μ, B, ϕ, q, etc).

8.1.1 Discretização das variáveis

As funções p e S_l serão aproximadas por combinações lineares

$$p^*(x, y, z, t) = \sum_{j=0}^{n-1} a_j(t)\varphi_j(x, y, z) \qquad e \qquad S_l^*(x, y, z, t) = \sum_{j=0}^{n-1} (b_l)_j(t)\varphi_j(x, y, z)$$

onde cada $\varphi_j(x,y,z)$ é uma função de base do espaço de aproximação, que depende apenas da posição (x,y,z), e os coeficientes $a_j(t)$ e $(b_l)_j(t)$ variam com o tempo. A solução será calculada apenas para valores discretos de tempo t.

8.1.2 Critério de Galerkin

A discretização das equações de escoamento (7.7) é feita pelo método de Galerkin (seção 5.2), aplicado às coordenadas espaciais para um instante t fixo, como no capítulo 6. Lembramos que este método exige que o resíduo das equações diferencias (7.7) seja ortogonal às funções da base. Obtemos então para cada fase l as equações

$$\int_\Omega \left(\nabla \cdot W_l - \frac{\partial}{\partial t}\left(\rho_l \frac{S_l^* \phi}{B_l}\right) + \rho_l q_l \right) \varphi_i dV = 0 \qquad i = 0, \ldots, n-1$$

onde $dV = dxdydz$, o símbolo '$\cdot$' indica o produto escalar no $\mathbb{R}^3$ e W_l é o campo vetorial

$$W_l = \rho_l K \frac{k_l}{\mu_l B_l}(\nabla p^* - \rho_l g)$$

Ou seja,

$$\int_\Omega (\nabla \cdot W_l)\varphi_i dV = \int_\Omega \frac{\partial}{\partial t}\left(\rho_l \frac{S_l^* \phi}{B_l}\right)\varphi_i dV - \int_\Omega \rho_l q_l \varphi_i dV \qquad i = 0, \ldots, n-1 \quad (8.1)$$

Observe que, de modo geral, o lado esquerdo da equação (8.1) mede o deslocamento dos fluidos, e o lado direito mede a variação local das saturações com o tempo.

8.1.3 Termo do fluxo

Desenvolveremos primeiro o lado esquerdo da igualdade (8.1) acima. Aplicando a redução de derivadas descrita na seção 5.3.1, reescrevemos esta fórmula como

$$\int_\Omega \varphi_i (\nabla \cdot W_l) dV = -\int_\Omega (\nabla \varphi_i) \cdot W_l dV$$

Expandindo o termo W_l, obtemos

$$-\int_\Omega (\nabla \varphi_i) \cdot W_l dV = -\int_\Omega (\nabla \varphi_i) \cdot \left(\rho_l K \frac{k_l}{\mu_l B_l} (\nabla \sum_{j=0}^{n-1} a_j \varphi_j - \rho_l g) \right) dV =$$

$$-\int_\Omega (\nabla\varphi_i)\cdot\left(\sum_{j=0}^{n-1} a_j(\rho_l K\frac{k_l}{\mu_l B_l})\nabla\varphi_j - (\rho_l K\frac{k_l}{\mu_l B_l})(\rho_l g)\right)dV =$$

$$-\int_\Omega \sum_{j=0}^{n-1} a_j\left(\nabla\varphi_i\cdot(\rho_l K\frac{k_l}{\mu_l B_l})\nabla\varphi_j\right)dV + \int_\Omega \nabla\varphi_i\cdot(\rho_l K\frac{k_l}{\mu_l B_l})(\rho_l g)dV =$$

$$-\sum_{j=0}^{n-1} a_j\int_\Omega \nabla\varphi_i\cdot(\rho_l K\frac{k_l}{\mu_l B_l})\nabla\varphi_j dV + \int_\Omega \nabla\varphi_i\cdot(\rho_l K\frac{k_l}{\mu_l B_l})(\rho_l g)dV \qquad (8.2)$$

Expandindo agora o lado direito da igualdade (8.1), obtemos

$$\int_\Omega \frac{\partial}{\partial t}\left(\frac{\rho_l\phi}{B_l}\sum_{j=0}^{n-1}(b_l)_j\varphi_j\right)\varphi_i dV - \int_\Omega \rho_l q_l\varphi_i dV \qquad (8.3)$$

Para simplificar estas equações, vamos supor que as variações de pressão dentro do reservatório têm efeito desprezível sobre a densidade dos fluidos e sobre a porosidade da rocha. Portanto podemos considerar

$$\frac{\partial}{\partial t}\left(\frac{\rho_l\phi}{B_l}\right) = 0 \qquad (8.4)$$

Como as funções φ não dependem do tempo t, a fórmula (8.3) fica

$$\sum_{j=0}^{n-1}\left(\frac{\partial}{\partial t}(b_l)_j\right)\int_\Omega \frac{\rho_l\phi}{B_l}\varphi_j\varphi_i dV - \int_\Omega \rho_l q_l\varphi_i dV \qquad (8.5)$$

8.2 Forma matricial das equações

A fórmula (8.2) do lado esquerdo da equação (8.1), na forma matricial, pode ser escrita como $E_l a + f_l$, onde a é o vetor (coluna) dos coeficientes da pressão p; E_l é a matriz $n \times n$ definida por

$$(E_l)_{i,j} = -\int_\Omega (\rho_l\frac{k_l}{\mu_l B_l})(\nabla\varphi_i)\cdot(K\nabla\varphi_j)dV \qquad i,j = 0,\ldots,n-1 \qquad (8.6)$$

e f_l é o vetor (coluna) definido por

$$(f_l)_i = \int_\Omega (\nabla \varphi_i) \cdot (\rho_l K \frac{k_l}{\mu_l B_l})(\rho_l g) dV \qquad i = 0, \ldots, n-1 \tag{8.7}$$

A fórmula (8.5) do lado direito, por sua vez, torna-se $G_l b_l' + h_l$, onde b_l' é o vetor (coluna) das derivadas dos coeficientes $(b_l)_j$ da saturação S_l, $(b_l')_j = \frac{\partial (b_l)_j}{\partial t}$; G_l é a matriz $n \times n$ definida por

$$(G_l)_{i,j} = \int_\Omega (\frac{\rho_l \phi}{B_l}) \varphi_j \varphi_i dV \qquad i, j = 0, \ldots, n-1 \tag{8.8}$$

e h_l é o vetor (coluna) definido por

$$(h_l)_i = -\int_\Omega \rho_l q_l \varphi_i dV \qquad i = 0, \ldots, n-1 \tag{8.9}$$

As equações (8.1), portanto, podem ser escritas como

$$E_l a + f_l = G_l b_l' + h_l \tag{8.10}$$

Colocaremos agora a saturação do óleo em função da saturação da água, usando a identidade $S_o + S_a = 1$. Sejam $c_1, \ldots, c_n$ tais que $\sum c_j \varphi_j(x) = 1$ para todo x em Ω. De $S_o = 1 - S_a$, concluímos que $(b_o)_j = c_j - (b_a)_j$ e portanto $(b_o')_j = -(b_a')_j$. Podemos então escrever as equações de fluxo destas duas fases como o sistema final

$$\begin{pmatrix} E_a & -G_a \\ E_o & -G_o \end{pmatrix} \begin{pmatrix} a \\ b_a' \end{pmatrix} = \begin{pmatrix} h_a & -f_a \\ h_o & -f_o \end{pmatrix} \tag{8.11}$$

Este sistema possui $2n$ equações, sendo n equações para cada fase, e $2n$ variáveis, que são os vetores a e b_a'.

8.3 Integração no tempo

A simulação do escoamento de petróleo consiste em resolver o sistema (8.11) para cada instante t fixo. Com isso obtemos os coeficientes das pressões a_j e as derivadas temporais $(b_a')_j$ da saturação da água. Usamos então as derivadas $(b_a')_j$ para calcular o estado do reservatório no instante seguinte, depois de um certo intervalo de tempo Δt, como no capítulo 6. Mais precisamente, as saturações para o próximo instante t são recalculadas pelo método de integração de Euler,

$$(b_a)_j^{(t+\Delta t)} \leftarrow (b_a)_j^{(t)} + \Delta t (b_a)_j^{\prime(t)} \qquad j = 0, \ldots, n-1 \tag{8.12}$$

Finalmente avançamos para o próximo instante ($t \leftarrow t + \Delta t$) e repetimos o processo todo.

8.3.1 Dados da simulação

No início do processo ($t = t_0$) fornecemos ao simulador a pressão típica $\bar{p}$ (constante ao longo da simulação) e a distribuição inicial da saturação da água S_a, descrita pelo vetor de coeficientes $b_a^{(0)}$. Supomos também que as taxas de injeção (as distribuições q_l) são conhecidas em cada instante t. A evolução do estado do reservatório ocorre devido à vazão dos poços ($q_l \neq 0$), gerando diferenças de pressão que por sua vez causam movimento do fluido, ou seja, variação nas saturações ao longo do tempo.

8.3.2 Cálculo das pressões e fluxos

Note que o sistema (8.11) não é linear, pois tanto as matrizes E_l e G_l quanto os vetores h_l e f_l dependem dos coeficientes a_j (eles dependem também dos coeficientes b_{aj}, mas estes são conhecidos num dado instante t). Portanto a solução do sistema é determinada através do método iterativo de Newton [PFTV86]. Usamos como aproximações iniciais para os coeficientes da pressão os mesmos coeficientes do instante anterior:

$$a_j^{(t)} \leftarrow a_j^{(t-\Delta t)} \qquad j = 0, \ldots, n-1$$

A cada iteração do método de Newton, recalculamos as matrizes E_l e G_l e os vetores h_l e f_l usando os valores mais recentes de a, e os valores de b_a (que são fixos para cada t). Em seguida resolvemos o sistema (8.11) supondo que essas matrizes e vetores são independentes de a.

$$\begin{pmatrix} a \\ b'_a \end{pmatrix} \leftarrow \begin{pmatrix} E_a & -G_a \\ E_o & -G_o \end{pmatrix}^{-1} \begin{pmatrix} h_a & -f_a \\ h_o & -f_o \end{pmatrix}$$

Efetuamos várias destas iterações até o vetor a convergir.

Note que a suposição de que os fluidos são incompressíveis (8.4) implica que a pressão p_l não tem 'memória', isto é, ela depende apenas das saturações e taxas de injeção nos poços q_l no instante t, e não da pressão nos instantes anteriores.

8.3.3 Cálculo das taxas de injeção

As funções q_l deveriam ser zero em todo o reservatório, exceto nas vizinhanças imediatas dos poços. Na nossa formulação, entretanto, estas funções são aproximadas por combinações $q_l = \sum e_{li}\varphi_i$ de tendas da base. Também adotamos

sempre a base $\mathcal{B}_{min}$, colocando os poços sobre vértices completos da grade, de maneira que nas combinações cada poço é representado por uma única tenda φ_i.

A taxa de injeção q_a normalmente é diferente de zero apenas na vizinhança dos poços onde ocorre injeção de água. Nestes pontos q_a é positivo. Entretanto, q_a pode assumir valores negativos em algum poço de extração se a saturação de água nesse ponto for positiva. A vazão q_o, por outro lado, em geral nunca é positiva porque não há injeção de óleo, e assume valores negativos somente nos poços de extração.

Na prática são dadas apenas as vazões totais Q_{lj} em cada poço, em função do tempo. Se a vazão da fase l no poço j é representada na taxa q_l pelo termo $e_{li}\varphi_i$, então devemos ter

$$Q_{lj} = \int_\Omega e_{li}\varphi_i dV$$

Na verdade para poços de injeção Q_{aj} é dado e Q_{oj} é sempre zero. Para poços de extração, conhecemos apenas a soma $Q_j = Q_{aj} + Q_{oj}$ (vazão total de fluido bombeado). A proporção de água e óleo no fluido obtido depende de suas saturações e mobilidades relativas das duas fases. Portanto, recalculamos as vazões em cada poço de extração j a cada instante de tempo usando a seguinte relação:

$$\frac{Q_{oj}}{m_{oj}} = \frac{Q_{aj}}{m_{aj}}$$

onde

$$m_{lj} = \int_\Omega \frac{k_l}{\mu_l B_l}\varphi_i dV$$

e φ_i é a tenda que representa o poço j [MD90].

8.4 Adaptação dinâmica da grade

O processo descrito na seção 8.3 supõe que a grade G é fixa durante toda a simulação. Para realizar o processo iterativo de integração com uma grade variável, é necessário acrescentar alguns passos adicionais entre um instante e outro de simulação.

Após calcularmos a saturação $(b_a)^{(t+\Delta t)}$ pela equação (8.12), e fazer $t = t+\Delta t$, procuramos uma nova grade diádica G', que seja mais adequada a representar o novo estado do reservatório. De posse de G', calculamos uma base $\mathcal{B}'$ para o espaço de *splines* $\mathcal{E}^1[G']$ correspondente, usando os algoritmos da seção 3.6 (Gera_Bmax ou Gera_Bmin).

8.4.1 A grade inicial G_{ini}

Existem diversos critérios que podemos usar para escolher a nova grade G'. Em nossa abordagem, a grade G' é sempre um refinamento de uma grade inicial G_{ini}.

A grade G_{ini}, por sua vez, é gerada no começo da simulação. Ela possui um refinamento máximo dado P_{max} na vizinhança dos poços e na fronteira do reservatório, pois estas são regiões críticas que não variam com o tempo. O tamanho das células de G_{ini} aumenta gradualmente com a distância a estas regiões. Supomos que na grade G_{ini} as propriedades petrofísicas estão representadas com um grau suficiente de detalhamento, o que garante que elas serão também adequadamente representadas em qualquer refinamento G' da mesma.

8.4.2 Escolha de uma nova grade

Escolhemos a nova grade G' executando o algoritmo Refina_Grade (4.5.1) sobre a saturação da água ($S_a^* = \sum_{j=0}^{n-1}(b_a)_j\varphi_j$), com a profundidade máxima dada P_{max}, em cada célula da grade fixa G_{ini}.

8.4.3 Mudança de base

Cada vez que a grade G' muda, calculamos os coeficientes $(b_a)_j$ da saturação da água $S_a^{*(t+\Delta t)}$ e a_j da pressão $p^{*(t+\Delta t)}$ na nova base $\mathcal{B}'$ através de aproximações por mínimos quadrados (seção 4.1) destas funções. Em particular, quando a nova grade G' é um refinamento da grade anterior G (ou seja, nenhuma célula de G foi reagrupada em G') então estas aproximações são iguais às funções originais.

No calculo das taxas de injeção $q_l = \sum e_{li}\varphi_i$ na nova base, temos que encontrar para cada poço j do reservatório o índice i da tenda φ_i de $\mathcal{B}'$ cujo centro é esse poço. Vale notar que estas tendas sempre existem e têm o mesmo suporte, pois G' e G_{ini} têm profundidade igual a P_{max} em volta de cada poço e portanto todos os poços são vértices completos de G_{ini} e G'.

8.4.4 Estratégia adaptativa

Com objetivo de manter a precisão dos dados, trabalhamos na verdade com duas malhas G' e G''. A *malha grossa* G' é obtida pelo algoritmo Refina_Grade como descrito na seção 8.4.2. A *malha fina* G'' é um refinamento de G', onde cada uma de suas células folha é dividida em até k níveis adicionais, sem exceder a profundidade máxima P_{max}. Este refinamento excedente da malha fina (supérfulo

em algumas regiões do reservatório) serve para manter a fidelidade do processo dinâmico.

Na versão atual do nosso simulador, a grade grossa G' é usada durante um intervalo fixo de tempo Δt_{gros}. Ao final desse intervalo, a grade fina G'' é construída. A grade fina G'' é usada durante um intervalo fixo de tempo Δt_{fin}. Após o término de Δt_{fin}, recalculamos a grade grossa G'.

Uma abordagem mais correta seria efetuar a simulação com as duas grades G' e G'', e comparar os resultados, procurando manter o refinamento mínimo que produz uma resposta semelhante nas duas.

8.5 Cálculo das matrizes e vetores do sistema

Vamos detalhar agora o cálculo das matrizes E_l, G_l e dos vetores f_l e h_l - equações (8.6) a (8.9).

Observe que as integrais que aparecem nas fórmulas das matrizes E_l e G_l, embora estejam definidas em todo o domínio Ω, podem ser calculadas apenas na interseção Ω_{ij} do suporte das tendas φ_i e φ_j, porque fora desta região o resultado da integração será sempre zero. O mesmo vale para as integrais presentes nos vetores f_l e h_l, que podem ser calculadas apenas no suporte Ω_i das tendas φ_i.

8.5.1 Decomposição em células

Nos dois casos, a região de integração consiste de zero ou mais células inteiras do nível $P = \max\{P(\varphi_i), P(\varphi_j)\}$. Podemos portanto decompor estas integrais em somatórias de integrais parciais, sobre cada uma das células do nível P, obtendo

$$\int_{\Omega_{ij}} f(r)\varphi_i(r)\varphi_j(r)dr = \sum_{C \in D_{ij}} \int_C f(r)\varphi_i(r)\varphi_j(r)dr$$

onde D_{ij} é o conjunto de células do nível P que compõem Ω_{ij}, ou

$$\int_{\Omega_i} f(r)\varphi_i(r)dr = \sum_{C \in D_i} \int_C f(r)\varphi_i(r)dr$$

onde D_i é o conjunto de células do nível P que compõem Ω_i.

8.5.2 Determinação dos termos constantes

Além disso, nas integrais vamos supor que as propriedades ρ, k, μ, B, ϕ e o tensor K são praticamente constantes no interior de cada célula C pertencente aos conjunto D_{ij} (região Ω_{ij}) ou D_i (região Ω_i), respectivamente. Assim, podemos fazer as seguintes simplificações

$$(E_l)_{i,j} = -\int_\Omega (\rho_l \frac{k_l}{\mu_l B_l})(\nabla\varphi_i)\cdot(K\nabla\varphi_j)dV \simeq \sum_{C\in D_{ij}} -\frac{\rho_l(r)K(r)k_l(r)}{\mu_l(r)B_l(r)}\int_C(\nabla\varphi_i\cdot\nabla\varphi_j)dV$$

tal que r é o ponto central de C, e

$$(f_l)_i = \int_\Omega (\nabla\varphi_i)\cdot(\rho_l K\frac{k_l}{\mu_l B_l})(\rho_l g)dV \simeq \sum_{C\in D_i}\frac{\rho_l(r)^2K(r)k_l(r)}{\mu_l(r)B_l(r)}\int_C(\nabla\varphi_i\cdot g)dV$$

A permeabilidade K é portanto considerada um escalar, e não um tensor. Aplicando o mesmo princípio à matriz G_l, temos

$$(G_l)_{i,j} = \int_\Omega (\frac{\rho_l\phi}{B_l})\varphi_j\varphi_i dV \simeq \sum_{C\in D_{ij}}\frac{\rho_l(r)\phi(r)}{B_l(r)}\int_C(\varphi_j\varphi_i)dV$$

No caso do vetor h_l, substituindo a taxa q_l pela combinação linear $\sum e_{lj}\varphi_j$, podemos escrever

$$(h_l)_i = -\int_\Omega \rho_l q_l\varphi_i dV = -\int_\Omega \rho_l(\sum e_{lj}\varphi_j)\varphi_i dV$$

Se tomarmos apenas a tenda φ_j cujo suporte possui interseção não nula com o suporte de φ_i, então chegamos a

$$(h_l)_i \simeq -\sum_{C\in D_{ij}}\rho_l(r)e_{lj}\int_C(\varphi_j\varphi_i)dV$$

Assim, é possível calcular as matrizes E_l, G_l e os vetores f_l e h_l integrando somente tendas e gradientes de tendas (sendo a aceleração da gravidade g invariável no domínio) e multiplicando os resultados pelos valores das propriedades tomados em pontos específicos do domínio de integração. Contudo, para um mesmo instante t, recalculamos estas matrizes e vetores a cada iteração na solução de (8.11), pois as propriedades ρ, μ e B dependem das pressões e variam conforme as mudanças no vetor a.

Capítulo 9

O simulador Simóleo

Descrevemos neste capítulo os módulos que compõem o simulador de escoamento de petróleo que desenvolvemos neste projeto, chamado *Simóleo*, e alguns experimentos que realizamos para testá-lo.

9.1 O Simóleo

9.1.1 Funcionalidades

As principais características do Simóleo são:

- *Modularidade*: Oferece conjuntos independentes de bibliotecas de suporte para (1) construção de grades diádicas, (2) determinação das bases $\mathcal{B}_{max}$ ou $\mathcal{B}_{min}$ de *splines*, (3) aproximação usando bases de *splines* e (4) visualização dos resultados. Cada uma destas bibliotecas pode ser substituída com relativamente pouco impacto;
- *Grades adaptativas*: Implementa a discretização e integração das equações de escoamento (capítulo 8), com grade estática ou adaptativa;
- *Dimensão variável*: O programa pode ser usado sobre domínios com 1, 2 ou 3 dimensões. As bibliotecas de grades diádicas e determinação de bases de *splines* têm esta mesma flexibilidade;
- Visualização gráfica *off-line* de resultados (no momento restrita a representações em 1 ou 2 dimensões).

9.1.2 Estrutura

O Simóleo consiste de um programa integrador propriamente dito, e vários programas auxiliares para geração de grades, bases, etc. Suas bibliotecas e programas foram criados em linguagem *C* padrão sobre plataforma *Unix*. Conceitualmente, estes programas podem ser divididos e reagrupados num conjunto de módulos funcionais com interfaces relativamente restritas e abstratas. Estes módulos são:

- **SPL:** geração e manipulação de grades diádicas, *splines* diádicos e respectivas bases;
- **EQS:** discretização e integração das equações de escoamento;
- **LIN:** álgebra de matrizes esparsas e resolução de sistemas lineares;
- **VIS:** visualização de grades e resultados.

ou, graficamente:

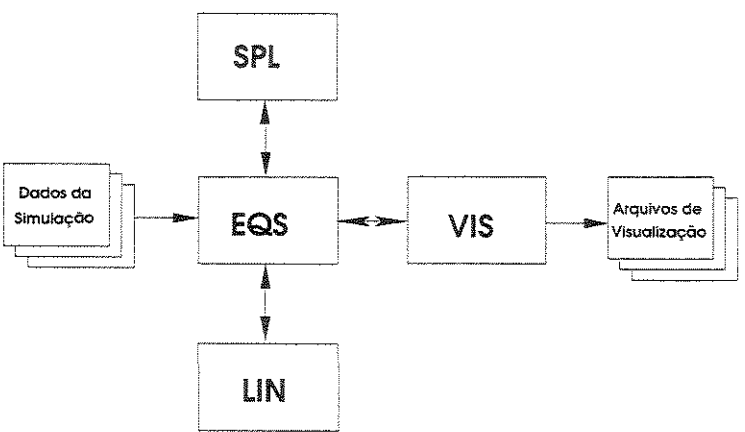


Figura 9.1: Módulos do programa Simóleo.

O módulo SPL

O módulo SPL é responsável pelo gerenciamento das grades e *splines* diádicos. Estes *splines* são formados por bases de tendas e combinações lineares destas bases. As grades diádicas são armazenadas como árvores binárias onde cada nó guarda um índice, dado por uma variável inteira. Cada tenda é representada pelo índice e profundidade de sua célula superior, também inteiros. Uma base é um

vetor destas estruturas. Um *spline* diádico genérico, por sua vez, é codificado como um vetor de coeficientes em ponto flutuante, referente a uma base especificada em separado.

Os algoritmos Interpola (3.5.1), Gera_Bmax/Gera_Bmin (3.6.1) e Refina_Grade (4.5.1) fazem parte do módulo SPL. Além disso, os procedimentos oferecidos por este módulo também incluem a manipulação das grades (inserção, remoção, busca), os produtos escalares entre tendas (seção 4.2) e entre o gradiente de tendas (seção 5.3.3).

O módulo EQS

Estão reunidas no módulo EQS as bibliotecas de discretização e integração (capítulo 8) das equações de escoamento multifásico óleo/água (capítulo 7). Estas bibliotecas utilizam as estruturas definidas no módulo SPL para representar os reservatórios e suas propriedades físicas. Cada reservatório é construído como uma grade diádica, refinada conforme a posição de seus poços, sua geometria e grau de detalhamento requerido. Suas propriedades mais importantes (saturações, pressões e taxas de fluxo) são guardadas como combinações lineares de tendas, e as demais (permeabilidades, viscosidades, etc) são codificadas como procedimentos.

Para realizar as integrações no espaço, os programas do módulo EQS montam sistemas lineares através do método iterativo de Newton que são resolvidos por rotinas providas pelo módulo LIN. Todas as propriedades petrofísicas podem ser visualizadas por métodos fornecidos pelo módulo VIS.

O módulo VIS

O módulo de visualização VIS inclui rotinas para plotar qualquer função do sistema. As imagens geradas por este módulo podem ser gráficos unidimensionais ou curvas de nível bidimensionais. Os gráficos unidimensionais servem para mostrar tanto funções de domínio com uma dimensão quanto cortes de funções k-dimensionais entre dois pontos especificados.

O módulo LIN

O módulo LIN consiste de rotinas para resolução de sistemas lineares. Na implementação atual, usamos apenas algoritmos simples baseados em eliminação de Gauss e fatoração de Cholesky [PFTV86].

Vale observar que as matrizes de produtos escalares de elementos finitos são esparsas. Portanto, para a eficiência do simulador, é indispensável utilizar algoritmos específicos para matrizes esparsas. Esta modificação deveria ser objeto de trabalho futuro.

9.2 Testes de simulação

Realizamos alguns experimentos com o objetivo de avaliar os efeitos dos refinamentos diádicos estático e dinâmico nas simulações. Para isso simulamos em duas dimensões um mesmo reservatório fictício utilizando três grades regulares, de níveis 6, 8 e 10, e uma grade irregular adaptativa dinâmica com profundidade máxima 10 ($P_{max} = 10$), obtida segundo o método descrito na seção 8.4. Todos os experimentos foram conduzidos em um computador contendo dois processadores AMD Athlon funcionando a uma frequência de $1GHz$, com $2G$ de memória, disco rígido *SCSI* com $40G$ e sistema operacional UNIX (Linux).

9.2.1 Dados de simulação

Os testes são baseados no modelo popular de cinco poços [PG00]. Usamos um reservatório fictício medindo $3,0\,km \times 3,0/\sqrt{2}\,km$ de área, com $1\,m$ de espessura, plano e sem nenhuma inclinação ($g_x = g_y = 0$). Há um poço de injeção no centro do domínio, no ponto $(\frac{3}{2}, \frac{3}{4}\sqrt{2})$, e quatro poços de extração equidistantes do poço de injeção nos pontos $(\frac{3}{4}, \frac{21}{16}\sqrt{2})$, $(\frac{9}{4}, \frac{21}{16}\sqrt{2})$, $(\frac{3}{4}, \frac{3}{16}\sqrt{2})$ e $(\frac{9}{4}, \frac{3}{16}\sqrt{2})$; todas as coordenadas em quilômetros. Veja a figura 9.2.

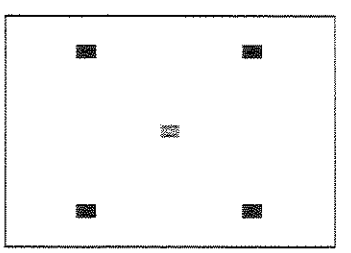


Figura 9.2: Disposição dos cinco poços no teste com o Simóleo.

Em todos os testes, o reservatório no início possui saturação da água uniforme $S_a = 0,1$. A pressão típica usada é $\bar{p} = 15000\,kPa$. O poço de injeção possui uma vazão de água constante $Q = 691\,m^3/\text{dia}$, e cada poço de extração tem uma

vazão total (água + óleo) $Q = 172,75\,m^3/\text{dia}$. As permeabilidades relativas k_l, viscosidades μ_l, e fatores volume de formação B_l das duas fases são dadas pelas equações:

$$\begin{aligned} k_a &= S_a^2 & k_o &= (1 - S_a)^2 \\ \mu_a &= 0,004 & \mu_o &= -2,6 \times 10^{-9} \times (p + \bar{p}) + 0,1 \\ B_a &= 1 & B_o &= 2,3 \times 10^{-8} \times (p + \bar{p}) + 1 \end{aligned}$$

sendo que a pressão é medida em Pa e a viscosidade em $Pa \times s^{-1}$. Consideramos as densidades constantes, com $\rho_a = 1000\,kg/m^3$ e $\rho_o = 887\,kg/m^3$.

A porosidade ϕ tem valor unitário em todo o domínio, com exceção da região limitada por uma margem de $150\,m$ a partir da fronteira. Nesta região ϕ decai de um para zero, proporcionalmente à distância da fronteira. A permeabilidade absoluta K, que juntamente com a porosidade depende apenas da posição espacial, é dada por $K = 5 \times 10^{-10}\phi$ (em m^2).

As simulações englobam os primeiros dois anos e cinco meses (888 dias) de funcionamento do reservatório. Usamos um passo de tempo fixo de $\Delta t = 4$ dias. Na simulação com adaptação dinâmica da grade adotamos os intervalos $\Delta t_{gros} = 20$ dias para a grade grossa G' e $\Delta t_{fin} = 20$ dias para a grade fina G''.

9.2.2 Resultados

A figura 9.3 mostra a saturação da água S_a em 5 estágios de simulação, desde o dia 8 até o dia 888, envolvendo as três grades regulares (níveis 6, 8 e 10) usadas nos testes. Na figura 9.4 estão representadas as mesmas funções da figura 9.3, mas na forma de gráficos unidimensionais da saturação da água entre o poço injetor e algum dos poços de extração (por simetria, a figura vale para todos eles). As figuras 9.5 e 9.6 mostram a pressão p de forma análoga às figuras 9.3 e 9.4.

As figuras 9.7 a 9.10 seguem o mesmo esquema das figuras 9.3 a 9.6, substituindo contudo os resultados obtidos através da grade regular de nível 6 pelos resultados provenientes da simulação com refinamento adaptativo e grade de profundidade máxima 10.

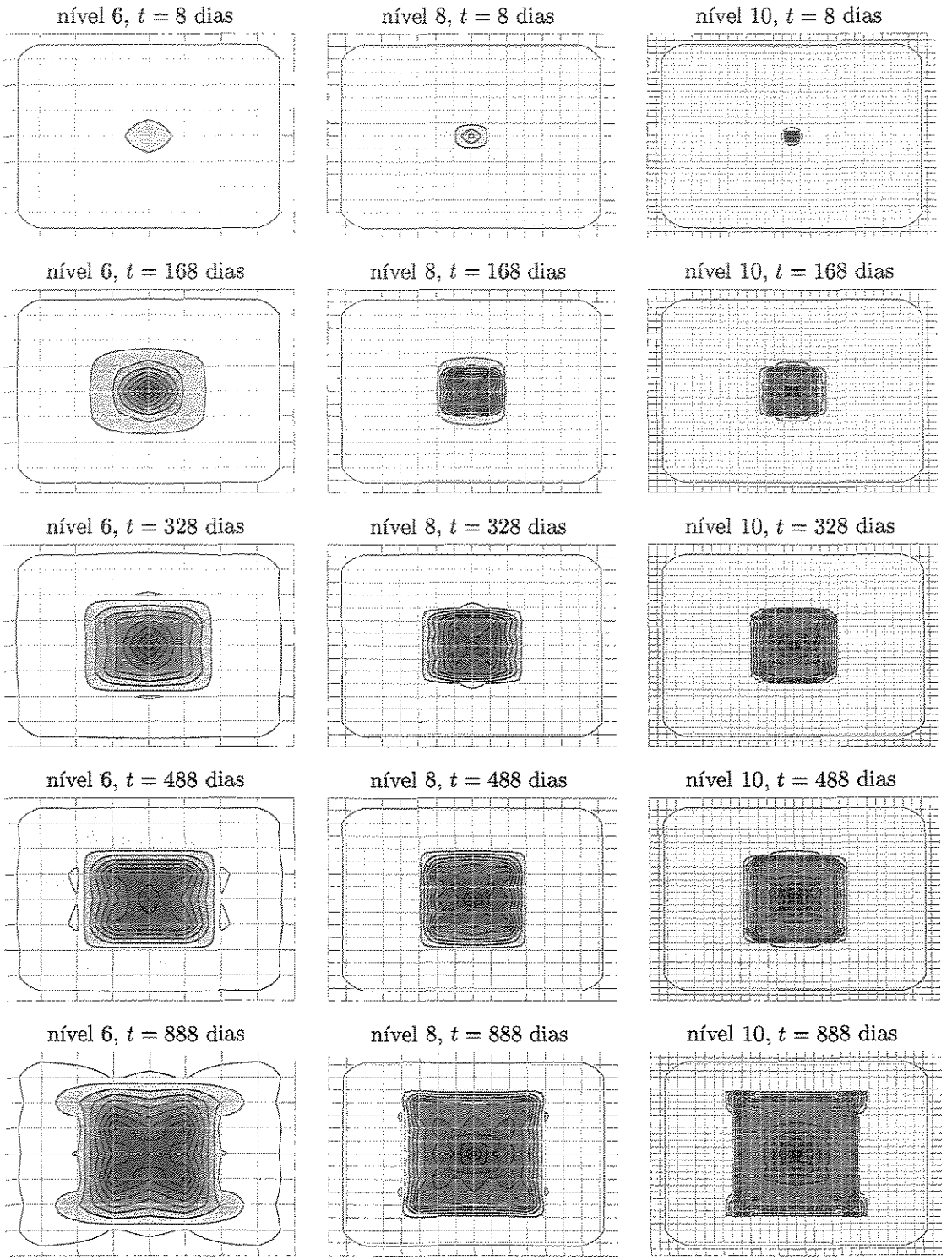


Figura 9.3: Saturação da água S_a em 5 estágios de simulação utilizando as grades regulares de níveis 6, 8 e 10. Todos os gráficos possuem $\delta_1 = 0,08$ e $\delta_2 = 1,0$.

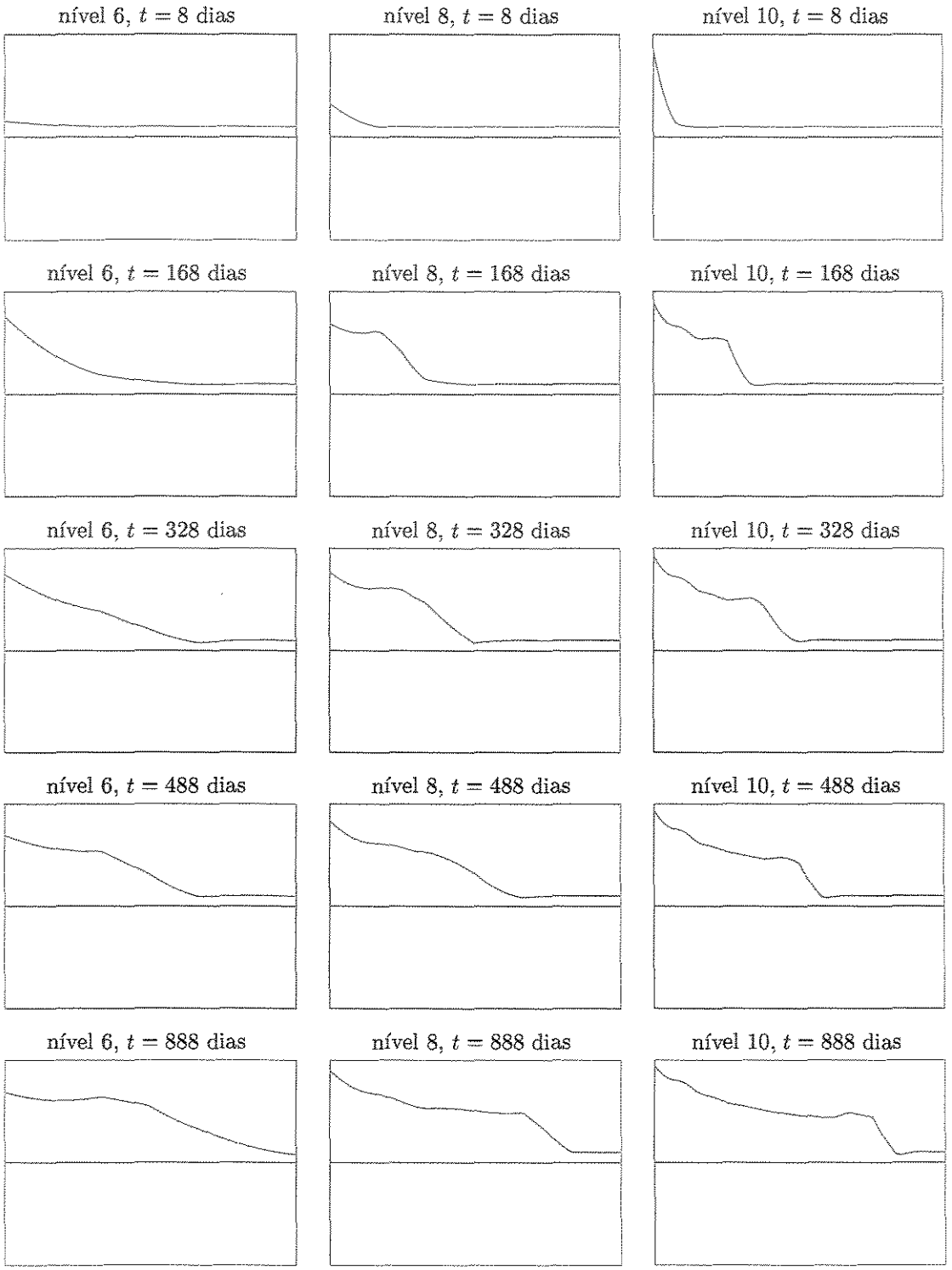


Figura 9.4: Perfis da saturação da água S_a sobre três grades regulares. Todos os gráficos possuem valor mínimo -1 e máximo 1 na escala vertical. A escala horizontal vai do poço injetor central até um poço de extração (1,2 km).

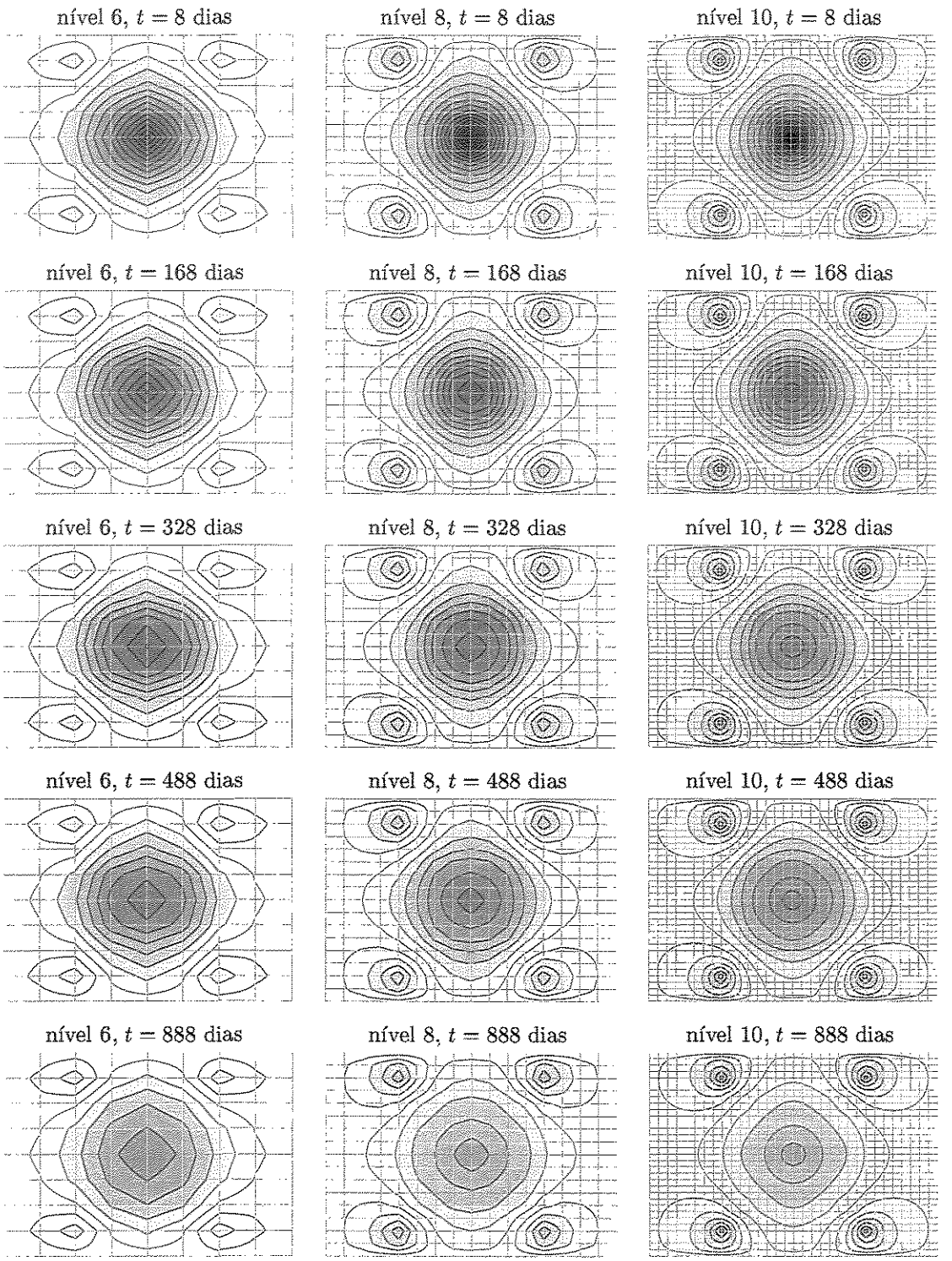


Figura 9.5: Pressão p em 5 estágios de simulação utilizando as grades regulares de níveis 6, 8 e 10, com $\delta_1 = 20000$ e $\delta_2 = 400000$. A pressão é positiva (acima de $\bar{p}$) em volta do poço de injeção, e negativa nos poços de extração.

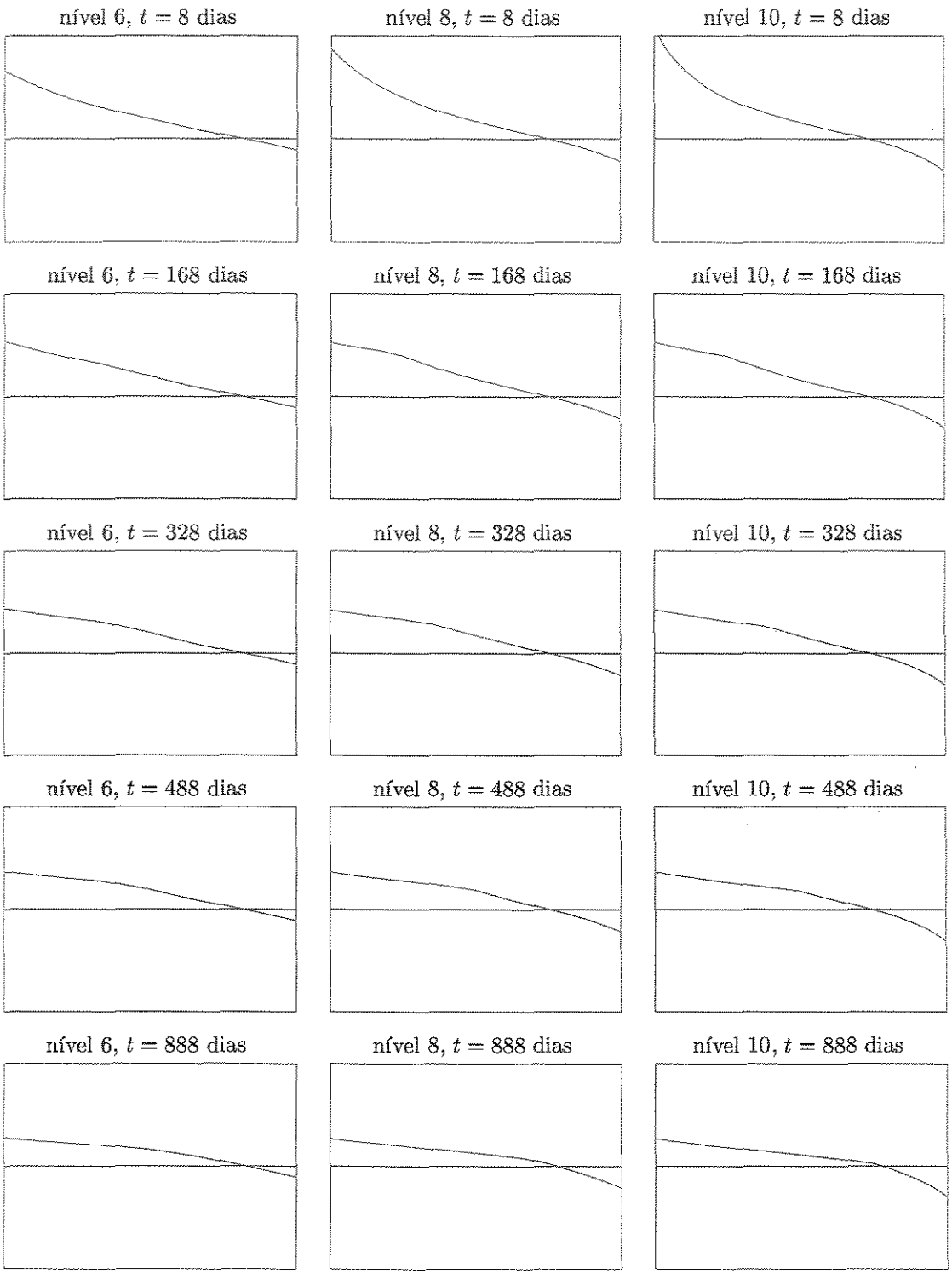


Figura 9.6: Perfis da pressão p sobre três grades regulares. A escala vertical varia de -400000 Pa a 400000 Pa. A escala horizontal vai do poço injetor central até um poço de extração (1,2 km).

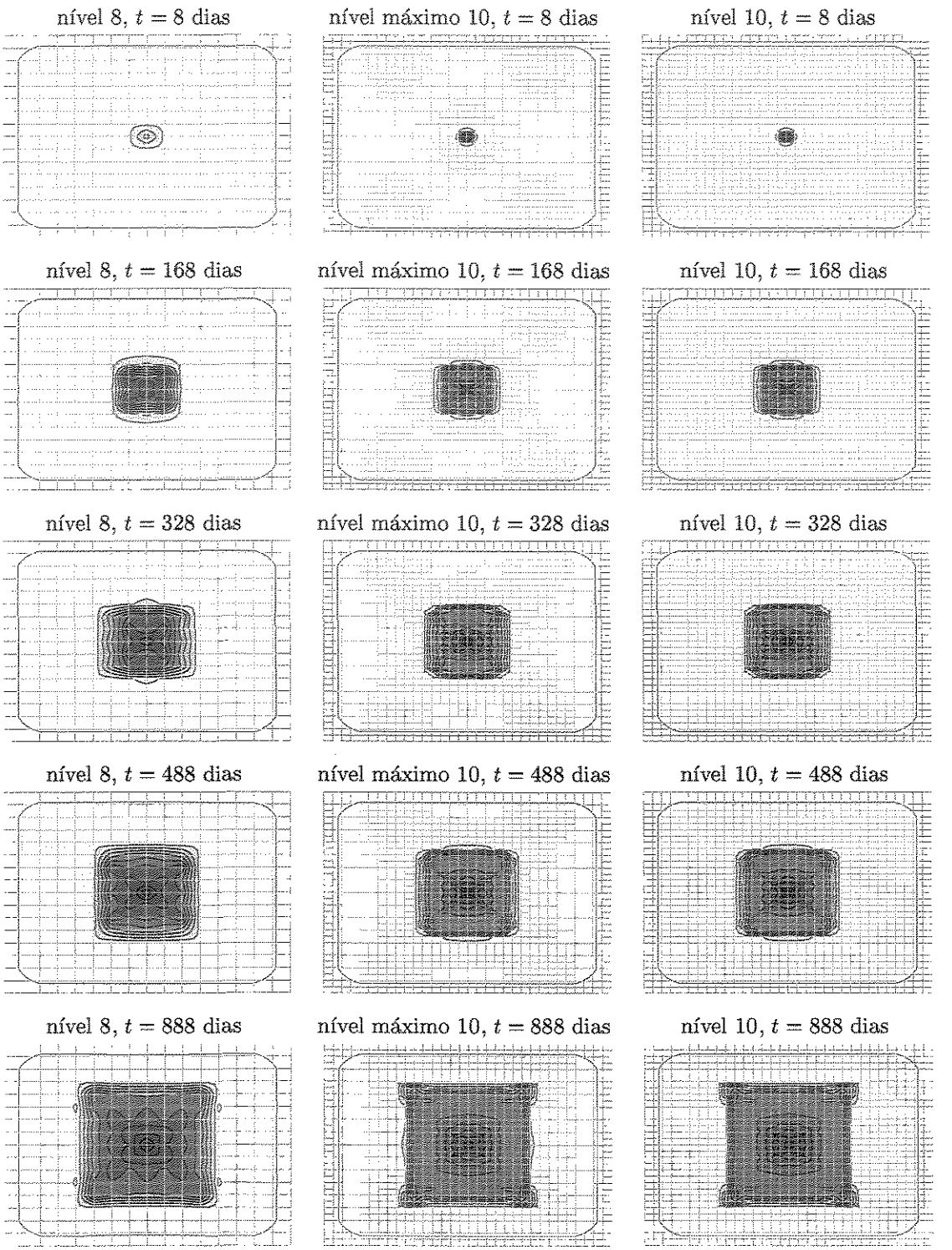


Figura 9.7: Saturação da água S_a sobre grades regulares de níveis 8, 10 e uma grade adaptativa dinâmica de nível máximo 10. Todos os gráficos possuem $\delta_1 = 0,08$ e $\delta_2 = 1,0$.

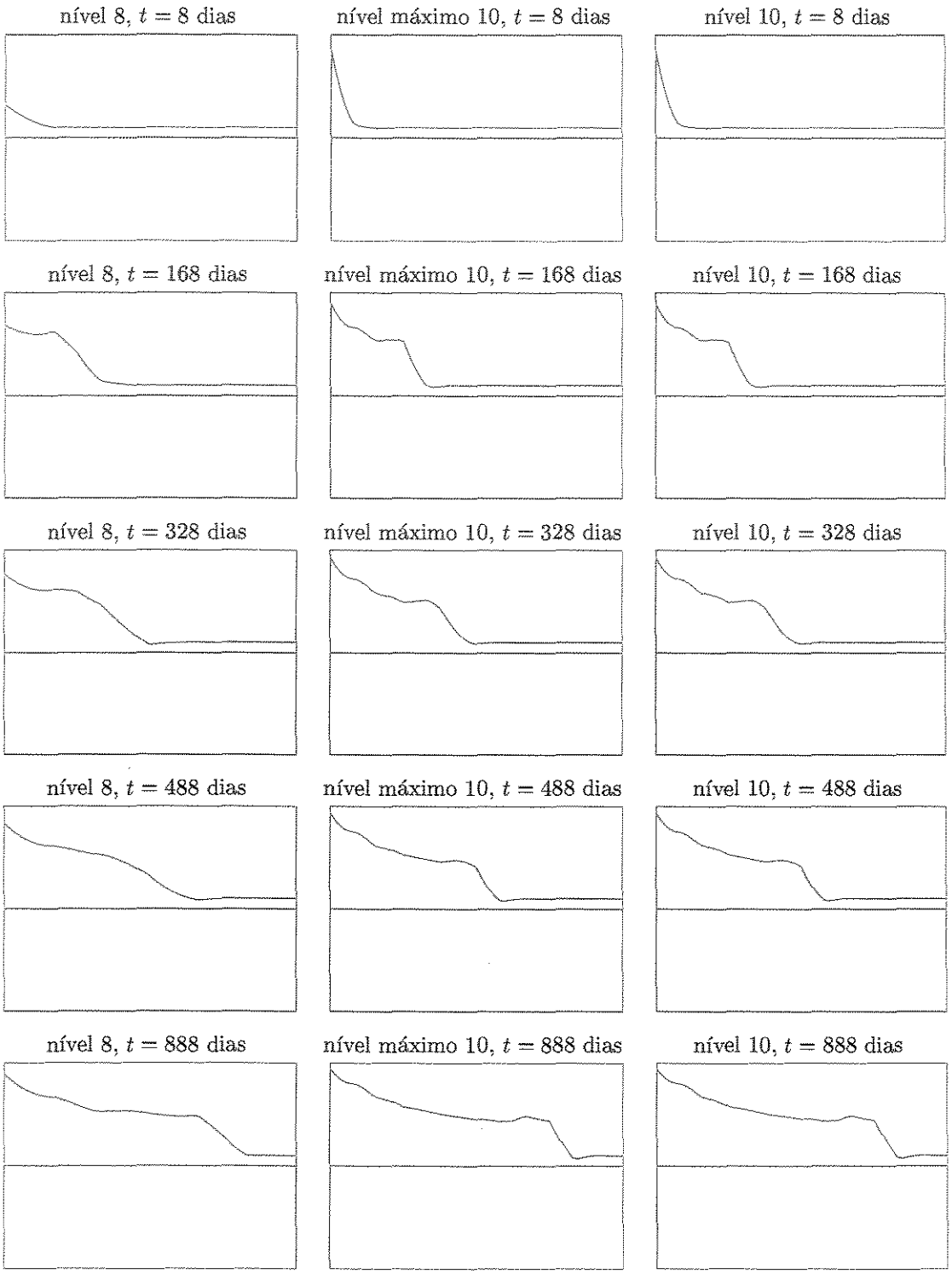


Figura 9.8: Perfis da saturação da água S_a sobre duas grades regulares e uma grade refinada adaptativamente, com valor mínimo -1 e máximo 1 na escala vertical. A escala horizontal vai do poço injetor central até um poço de extração (1,2 km).

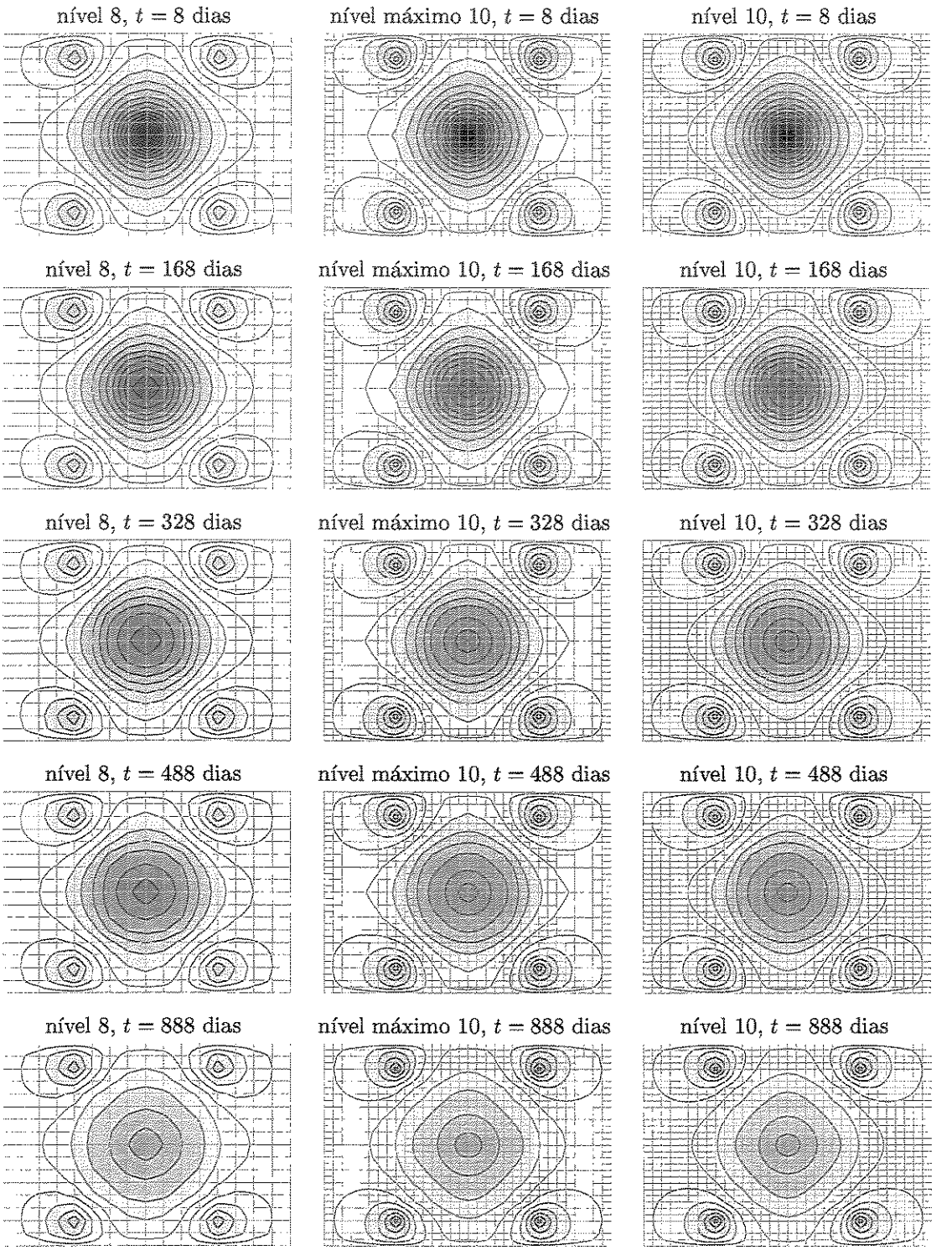


Figura 9.9: Pressão p utilizando grades regulares de níveis 8, 10, e uma grade com refinamento local dinâmico. Gráficos traçados com $\delta_1 = 20000$ e $\delta_2 = 400000$.

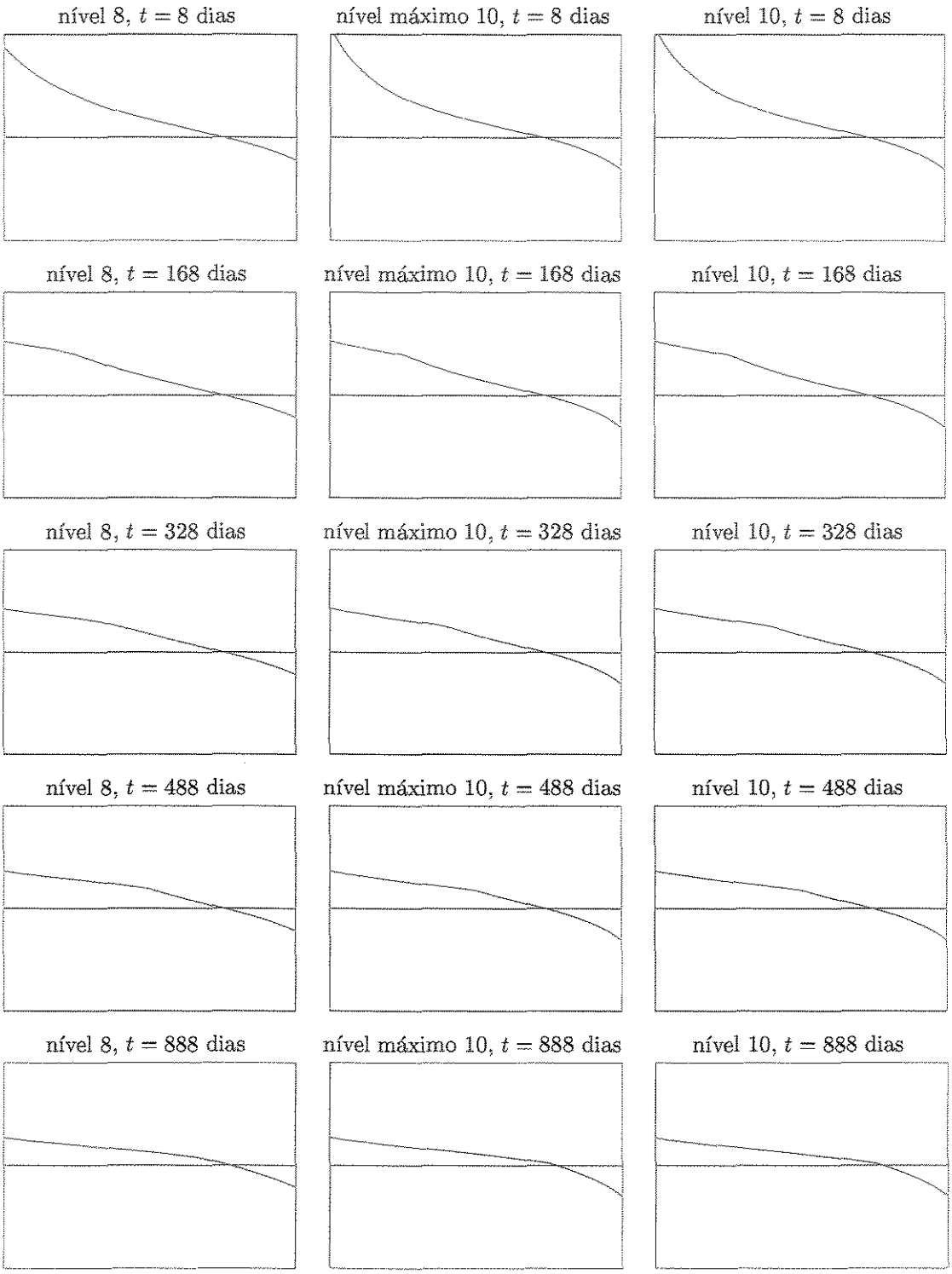


Figura 9.10: Perfis da pressão p sobre duas grades regulares e uma grade com refinamento local gerada dinamicamente. A escala vertical tem valor mínimo de -400000 Pa e máximo de 400000 Pa. A escala horizontal vai do poço injetor central até um poço de extração (1,2 km).

9.2.3 Evolução da pressão média

No gráfico 9.11 mostramos a pressão média do reservatório durante as simulações.

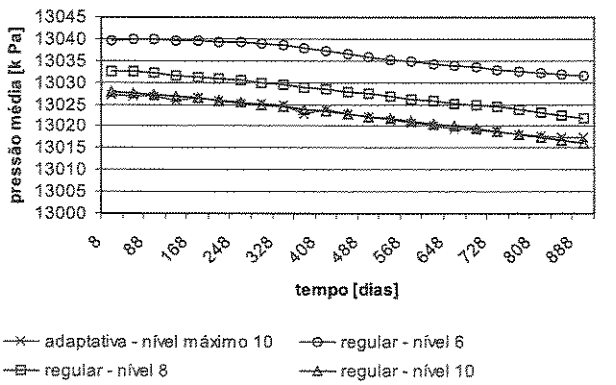


Figura 9.11: Pressão média do reservatório (em kPa) por tempo (em dias), nas simulações com grades regulares de níveis 6, 8, 10 e com a grade adaptativa de profundidade máxima 10.

9.2.4 Produção de óleo e água

A figura 9.12 mostra a produção total de óleo e água nos poços de extração. Observe que com a grade regular de nível 6, os poços começam a produzir água mais cedo, porque a frente de saturação desta fase é mais gradual (figura 9.4) e a área efetiva de cada poço é maior.

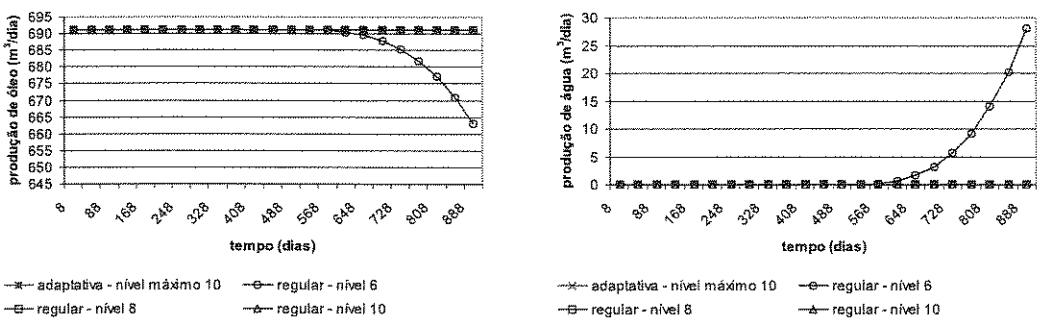


Figura 9.12: Taxas de produção de óleo (esquerda) e água (direita) em m^3/dia, nas simulações com grades regulares e com a grade refinada dinamicamente.

9.2.5 Tempo de simulação

A tabela 9.1 mostra o tempo total de cada simulação, em minutos.

nível	6	8	máximo 10	10
dimensão	49	225	467 – 739	961
tempo (min)	18	96	710	1131

Tabela 9.1: Tempo total em minutos das simulações com as grades regulares e com a grade adaptativa.

No teste com a grade adaptativa dinâmica, devido ao tamanho limitado do reservatório, não houve "engrossamento" da grade em nenhum estágio de simulação; ou seja, a cada iteração a grade era igual à anterior, ou mais refinada. Esta não é uma limitação do programa, mas apenas dos testes realizados.

9.2.6 Análise dos resultados

Analisando os gráficos apresentados podemos concluir que o refinamento da grade diádica gera uma melhora substancial na qualidade das simulações até o nível 10. A figura 9.4, em particular, mostra que a frente de saturação da água fica bem abrupta à medida que a resolução da grade aumenta. Este fato também é demonstrado pelos gráficos de produção mostrados na figura 9.12. O tempo total de simulação (tabela 9.1) aumentou bastante com o uso de grades mais finas.

Por outro lado, nos testes com refinamento adaptativo, obtivemos um resultado praticamente igual ao da simulação com a grade regular de nível 10, porém com aproximadamente 63% do tempo correspondente de execução.

Capítulo 10

Conclusões e trabalhos futuros

Neste capítulo relacionamos as principais conclusões deste trabalho, e sugerimos direções para pesquisas futuras.

10.1 Conclusões

10.1.1 Simplicidade de grades e bases diádicas

Conforme exposto nos capítulos 2 e 3, verificamos que as grades diádicas em qualquer dimensão podem ser criadas e manipuladas com facilidade. A estrutura regular das malhas permite uma representação muito simples, baseada em árvores binárias. Cada célula pode ser convenientemente especificada por um índice inteiro que determina completamente sua posição, profundidade e geometria. Isto nos permite construir algoritmos simples e eficientes para determinar as bases padrões $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$ de elementos finitos (funções tenda) do espaço $\mathcal{E}^1[G]$ associado a uma dada grade G. A estrutura regular das grades também nos permite pré-calcular os produtos escalares envolvendo estas funções.

10.1.2 Aproximação com *splines* diádicos

Através de testes numéricos (capítulos 4, 5 e 6) constatamos que os espaços $\mathcal{E}^1[G]$ são bastante eficazes na aproximação de funções e solução de equações diferenciais.

10.1.3 Grades adaptativas

Por meio de testes concluímos também que o refinamento adaptativo da grade melhora a precisão dos resultados sem aumentar exageradamente a dimensão dos espaços de aproximação, reduzindo portanto o tempo de execução dos experimentos.

Na simulação do escoamento multifásico de petróleo (capítulo 9), comprovamos as vantagens do refinamento dinâmico adaptativo da grade diádica. No teste realizado com este recurso, obtivemos dados com precisão equivalente aos da simulação feita com a grade regular mais fina, consumindo aproximadamente metade do tempo de execução gasto. Este resultado foi considerado bastante satisfatório, tendo em vista o custo adicional do refinamento dinâmico, que envolve a geração de novas grades e mudança de base das funções aproximadas.

10.2 Propostas futuras

10.2.1 *Splines* de graus mais elevados

Uma sugestão clara para trabalhos futuros é pesquisar os espaços de *splines* diádicos com graus e ordens de continuidade superiores às dos espaço $\mathcal{E}^1[G]$. Nestes espaços ($\mathcal{E}^2[G]$, $\mathcal{E}^3[G]$, etc) é provável que a relação entre a qualidade das aproximações e a necessidade de refinamento diádico seja melhor do que em $\mathcal{E}^1[G]$. Entretanto, precisaríamos avaliar se este ganho compensaria o custo de trabalharmos com funções de base mais complexas.

10.2.2 Escolha da base

Outra opção de trabalho futuro é realizar um estudo comparativo mais detalhado entre as bases padrões $\mathcal{B}_{max}$ e $\mathcal{B}_{min}$. O primeiro passo neste sentido é verificar a velocidade de convergência da solução iterativa de equações diferenciais lineares usando estas duas bases. Devido ao seu caráter multiescala, espera-se que a base $\mathcal{B}_{max}$ apresente convergência mais rápida.

10.2.3 Passo de tempo adaptativo

Na solução de equações diferenciais lineares dependentes do tempo, com a técnica descrita no capítulo 6, é de grande importância que possamos determinar valores

apropriados do passo de tempo Δt usado na integração de Euler. Simulações com valores muito grandes de Δt geram resultados imprecisos ou divergentes, ao passo que valores muito pequenos de Δt ocasionam atrasos desnecessários de processamento. Além disso, uma boa opção é ter um passo de tempo variável de acordo com a estrutura da grade G e o resultado das integrações espaciais.

10.2.4 Novas estratégias para integração no tempo

Um ponto interessante seria pesquisar estratégias mais sofisticadas de refinamento local dinâmico, para serem aplicadas na solução de equações diferenciais dependentes do tempo (capítulo 6). A princípio, isto englobaria a criação de novos algoritmos para a obtenção de grades adaptativas, em substituição ao procedimento Refina_Grade (4.5.1), e métodos mais eficientes para fazer a mudança de base das funções aproximadas.

10.2.5 Sugestões para o Simóleo

Entre os próximos testes com o programa Simóleo, devem ser incluídas simulações de reservatórios com variações na escala vertical (domínio tridimensional) e em regiões inclinadas ($g_x \neq 0$ ou $g_y \neq 0$). Para visualizar o resultado destes testes, entretanto, novas rotinas deveriam ser incorporadas ao módulo VIS (seção 9.1.2) para criação de gráficos 3D.

Com relação aos sistemas numéricos, é preciso adicionar ao módulo LIN (seção 9.1.2) rotinas com métodos iterativos para solução de matrizes grandes de sistemas lineares, bem como matrizes esparsas destes sistemas. Também deve ser realizado um estudo detalhado para tratar os problemas de instabilidade numérica do modelo discreto adotado (capítulo 8), especialmente considerando as diferenças de escala das propriedades petrofísicas e malhas diádicas.

Por fim, deveria ser adicionada ao simulador a fase gás, também presente no modelo físico *Black-Oil*.

Referências Bibliográficas

[AS79] K. Aziz e A. Settari. *Petroleum Reservoir Simulation.* Applied Science, 1979.

[ATL91] H. S. Al-Towailib e J. S. Liu. The application of local grid refinement to simulate a large hydrocarbon reservoir as an alternative to a two-model approach. Em *SPE Middle East Oil Show, Bahrain*, number paper SPE 21392, Novembro 1991.

[BBB87] H. Bartels, J. C. Beatty, e B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann, 1987.

[BQDY89] S. Bu-Qing e L. Ding-Yuan. *Computational Geometry Curve and Surface Modeling.* Academic Press, 1989.

[Bra78] R. N. Bracewell. *The Fourier Transform and Its Applications.* McGraw-Hill, 1978.

[Cri77] H. B. Crichlow. *Modern Reservoir Engineering - A Simulation Approach.* Prentice-Hall, 1977.

[dCS98] C. de C. Santos. Modelo para análise de deslocamento miscível no meio poroso usando a teoria dos canais de fluxo. Dissertação de Mestrado, UNICAMP - Universidade Estadual de Campinas, Brasil, 1998.

[Duc96] M. A. Duchaineau. *Dyadic Splines.* Tese de Doutorado, University of California, Davis, 1996.

[GE97] E. N. Gourley e T. Ertekin. Application of a local grid refinement technique to model impermeable barriers in reservoir simulation. Em *1997 SPE Eastern Regional Meeting, Laxington, KY*, number paper SPE 39216, Outubro 1997.

[Gom99] A. Gomide. *Splines Polinomiais Não Homogêneos na Esfera.* Tese de Doutorado, UNICAMP - Universidade Estadual de Campinas, Brasil, 1999.

[Gue98] S. S. Guedes. *Uma abordagem multiescala na simulação numérica de reservatórios.* Tese de Doutorado, UNICAMP - Universidade Estadual de Campinas, Brasil, 1998.

[HGH83] Z. E. Heinemann, G. Gerken, e G. V. Hantelmann. Using local grid refinement in a multiple-application reservoir simulator. Em *Reservoir Simulation Symposium, San Francisco, CA*, number paper SPE 12255, Novembro 1983.

[HS99] G. R. Hjaltason e H. Samet. Improved bulk-loading algorithms for quadtrees. Em *7th International ACM Workshop on Advances in Geographic Information Systems, Kansas City, MO*, number paper SPE 20740, Novembro 1999.

[Joh87] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Cambridge University Press, 1987.

[Lei94] L. Leithold. *O Cálculo com Geometria Analítica.* Editora Harbra, 1994.

[MD90] C. Mattax e R. L. Dalton. *Reservoir Simulation AIME.* Society of Petroleum Engineers, 1990.

[Men77] G. P. Menzala. *Introdução às Equações Diferenciais Parciais.* IMPA, 1977.

[Mor85] M. E. Mortenson. *Geometric Modeling.* John Wiley and Sons, 1985.

[NLP+90] E. C. Nacul, C. Leprete, O. A. Jr. Pedrosa, P. Girard, e K. Aziz. Efficient use of domain decomposition and local grid refinement in reservoir simulations. Em *65th Annual Technical Conference and Exhibition of the Society of Petroleum Engineers, New Orleans, LA*, number paper SPE 20740, Setembro 1990.

[PFTV86] W. H. Press, B. P. Flannery, S. A. Teukolsky, e W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, 1986.

[PG00] K. Pruess e J. Garcia. A systematic approach to local grid refinement in geothermal reservoir simulation. Em *Proceedings World Geothermal Congress 2000, Kyushu - Tohoku, Japão*, Maio 2000.

[Pis80] N. Piskunov. *Differential and Integral Calculus.* Mir Publishers, 1980.

[Ris02] V. F. Risso. Simulação numérica de fluxo em regiões de reservatórios de petróleo com refinamento local e fronteiras abertas. Dissertação de Mestrado, UNICAMP - Universidade Estadual de Campinas, Brasil, 2002.

[Sew88] G. Sewell. *The Numerical Solution of Ordinary and Partial Differential Equations.* Academic Press, 1988.

[Str88] T. D. Streltsova. *Well Testing in Heterogeneous Formations.* Exxon, 1988.

[Twi84] E. H. Twizell. *Computational Methods for Partial Differential Equations.* Ellis Horwood Limited, 1984.

[Vic81] R. Vichnevetsky. *Computational Methods for Partial Differential Equations.* Prentice-Hall, 1981.

[Vve93] D. Vvedensky. *Partial Differential Equations with Mathematica.* Addison-Wesley, 1993.

[Was87] M. L. Wasserman. Local grid refinement for three-dimensional simulators. Em *Ninth SPE Symposium on Reservoir Simulation, San Antonio, Texas*, number paper SPE 16013, Fevereiro 1987.