Departamento de Ciências e Tecnologia
Mestrado em Estatística, Matemática e Computação

Método Implícito para Filtragem de Ruído em Imagem Médica por Difusão Complexa

Marlon de Matos de Oliveira

Março 2014

Universidade Aberta
Departamento de Ciências e Tecnologia
Mestrado em Estatística, Matemática e Computação

Método Implícito para Filtragem de Ruído em Imagem Médica por Difusão Complexa

Marlon de Matos de Oliveira

Dissertação apresentada na Universidade Aberta
para obtenção do grau de mestre em
Matemática, Estatística e Computação

Orientador: Prof. Dr. Pedro Serranho

Março 2014

Resumo

Com o intuito de auxiliar o tratamento de imagens, neste trabalho de dissertação será focado o desenvolvimento e implementação de um esquema implícito de diferenças finitas para resolver uma equação diferencial complexa de difusão, para aplicação em filtragem de ruído em imagens. Serão apresentados também exemplos de imagens reais de imagiologia médica, nomeadamente em imagens in vivo por tomografia de coerência óptica (OCT - optical coherence tomography) da retina humana. Além do método implicito, serão focados também alguns aspectos do método semi-implícito. Basicamente serão apresentados três algoritmos para difusão complexa, sendo dois do método semi-implícito - um com condição de fronteira de Dirichlet e outro com condição de fronteira de Neumann - e outro para o método totalmente implícito com condição de Dirichlet na fronteira. Por fim, serão ilustrados os resultados das aplicações desses filtros em imagens de testes e com as imagens reais OCT.

Palavras-chaves: difusão complexa, método implícito, diferenças finitas, filtro de ruído, imagens OCT

Abstract

In this work we focus on the development and implementation of an implicit finite difference scheme for solving a complex differential diffusion equation with application to noise filtering in images, with some examples of real images from medical imaging, in particular in vivo images by optical coherence tomography (OCT) of the human retina. In addition, some aspects of the semi-implicit method are mentioned. Basically, three algorithms for complex diffusion are presented, two semi-implicit methods, with Dirichlet boundary condition and Neumann boundary condition, and the last one for the fully implicit method with the Dirichlet boundary condition. Finally, the results illustrate applications of these filters in test images and OCT real images.

Keywords: complex diffusion, implicit method, finite difference, noise filter, OCT images

Agradecimentos

Primeiramente eu gostaria de agradecer a Deus, pois sem Ele, nada nesse mundo é possível. A Deus sim, por ter colocado pessoas fantásticas no meu caminho que me ajudaram a chegar até aqui:

- Ao meu professor orientador Pedro Serranho, sem ele nada dessa história seria como foi. Obrigado por sugerir o tema da dissertação e me suportar a cada etapa da criação deste trabalho, de me orientar tão de perto, mesmo estando longe.
- A colega de mestrado Elisa, que mesmo sem me conhecer pessoalmente sempre me ajudou com muitas dúvidas matemáticas. A amiga Luana Müller, que me ajudou na correção e ao amigo Fleury Filho, que algumas vezes abraçou a causa e não mediu esforços para me explicar fórmulas e teorias. Ainda agradeço a esses pela fonte de inspiração para os meus estudos.
- A Associação para a Investigação Biomédica e Inovação em Luz e Imagem que gentilmente cedeu as imagens médicas.
- A Paula, que durante o mestrado passou de namorada a noiva e agora esposa, sempre me apoiando em todos os momentos, me dando força e coragem para seguir a caminhada.
- A minha Mãe, que às vezes nem sabe direito que eu estou estudando ou fazendo, mas que sempre me apoia nos meus planos mais malucos e que longe ou perto sempre tem palavras confortantes e de apoio.
- Ao meu Pai (em memória), pelo exemplo.

"O fato é que eu caminho porque tenho coração e é ele quem dá a direção. Sempre que busco o silêncio e me volto pra mim, sinto um sussurrar em meus ouvidos dando a direção. E eu vou."

Mhanoel Mendes

Sumário

Lista de Figuras

Capítulo 1

Introdução

O dia a dia da atividade médica é marcado por uma busca constante de um diagnóstico preciso e da avaliação terapêutica. Para esse fim, o médico serve-se de uma grande variedade de técnicas de produção de imagens, entre elas destacam-se os métodos radiológicos (raios X, imagens de medicina nuclear, tomografias, etc.). De entre as tomografias, destaca-se a de coerência óptica (OCT - optical coherence tomography) utilizada para a imagem do fundo ocular humano, que possui muitos ruídos, o que atrapalha um diagnóstico preciso por parte do médico.

Partindo desse problema, é focado neste trabalho o desenvolvimento e implementação de um esquema implícito de diferenças finitas, para resolver uma equação diferencial complexa de difusão, para aplicação em filtragem de ruído dessas imagens. Além das imagens de teste, serão apresentados também exemplos de aplicação de imagens reais de imagiologia médica, nomeadamente em imagens in vivo por OCT da retina humana.

Além do método implícito, serão exibidos também alguns aspectos do método semi-implícito. Basicamente serão apresentados algoritmos para difusão complexa, sendo dois do método semi-implícito, para duas condições de fronteira diferentes, e outro para o método totalmente implícito com uma condição de fronteira. Por fim, serão ilustrados os resultados das aplicações desses filtros.

1.1 Motivação e Objetivos

Existem vários métodos matemáticos para filtragem de imagem médica, como por exemplo, a utilização de um filtro passo-a-baixo. Estes filtros são poucos conservadores em transições rápidas da imagem, levando a uma atenuação de transições rápidas de cor ou contraste na imagem, ou seja, a um efeito de *bluring*[1] da imagem. Existem algumas alternativas para compensar este processo indesejável, por exemplo, a aplicação de métodos de difusão complexa com penalização sobre os limites de diferentes objetos na imagem. Perona e Malik [Perona and Malik, 1990], introduziram o modelo de difusão

$$\frac{\partial I}{\partial t} = \mathrm{div}(D\nabla I) \tag{1.1.1}$$

em que $I = I(x,t)$ é a intensidade da imagem no ponto x após t tempo de difusão, e o coeficiente de difusão D é dado por

$$D = \frac{1}{(1+|\nabla I|^2)} \tag{1.1.2}$$

e ∇ representa o operador gradiente e logo *div* representa o operador divergência. Note-se que sobre uma transição rápida da imagem, tem-se que o gradiente da intensidade é muito elevado, tornando o coeficiente de difusão muito próximo de zero. Assim, a difusão sobre transições rápidas na imagem é fortemente penalizada, eliminando o efeito de bluring da mesma. Mais tarde, este coeficiente de difusão foi adaptado para

$$D = \frac{1}{(1+|\Delta I|^2)} \tag{1.1.3}$$

uma vez que nesta nova formulação, em que Δ representa o laplaciano, além de se manter uma penalização sobre transições rápidas, existe uma diferenciação entre o início e final da transição e o ponto de inflexão. Qualquer das definições anteriores têm um problema: Em estágios iniciais em que a imagem I tem muito ruído, o cálculo do seu gradiente ou laplaciano é muito instável. Assim, Gilboa [Gilboa et al., 2004] sugeriu que se considerasse difusão com um termo difusivo complexo

$$D = \frac{e^{i\vartheta}}{1+\left(\frac{Im(I)}{\kappa\vartheta}\right)^2}. \tag{1.1.4}$$

[1] Efeito de desfocagem.

com o argumento ϑ próximo de zero. Para a difusão com o coeficiente anterior tem-se

$$\lim_{\vartheta \to 0} \frac{Im(I(x,t))}{t\vartheta} = G * \Delta I(x,0) \tag{1.1.5}$$

em que G é uma gaussiana, ou seja, a convolução com G representa um filtro passo-a-baixo. Assim, o coeficiente (1.1.4) também penaliza a difusão em transições rápidas de intensidade, sem necessitar de cálculo adicional de derivadas. A aplicação deste filtro a imagens de OCT foi já obteve sucesso em [Bernardes et al., 2010, Salinas and Fernández, 2007]. Em ambos os casos, a equação diferencial foi discretizada por um método de diferenças finitas explícito, o que implica que o passo temporal é condicionado por condições de estabilidade [Araújo et al., 2013]. Assim, são necessárias bastantes iterações do método para atingir um tempo de difusão satisfatório.

1.2 Trabalho Proposto

O trabalho proposto passa por implementar um método de diferenças finitas implícito com o objetivo de tratar o problema exposto de difusão complexa para filtragem de imagens com ruído, e em particular, imagens OCT da retina humana. Uma vez que o método implícito é incondicionalmente estável [Araújo et al., 2013], espera-se reduzir substancialmente o número de iterações necessárias por este método. No entanto, a cada passo será necessário resolver um sistema de equações, o que aumenta a complexidade da implementação numérica do problema. Em particular, uma vez que a equação diferencial não é linear na solução I, será necessário aplicar um método para equações não lineares a cada passo do método implícito. Neste trabalho será utilizado o método de Newton para esta linearização.

Capítulo 2

Diferenças Finitas para Difusão Complexa

Os processos de difusão são comumente utilizados para processamento geral de imagens, tais como *impaiting*[1], suavização de imagens e visão estérea [Bernardes et al., 2010,Brox et al., 2004, Grossauer and Scherzer, 2003, Perona and Malik, 1990, Salinas and Fernández, 2007, Weickert, 1994, Weickert, 1997, Zimmer et al., 2008]. Aplicações particulares para suavização de imagens e redução de ruídos são de interesse [Bernardes et al., 2010, Gilboa et al., 2004, Perona and Malik, 1990, Salinas and Fernández, 2007] e foram largamente usadas na última década. Os seguintes métodos são geralmente baseados na diferenciação de uma equação de difusão não linear da forma

$$\frac{\partial u}{\partial t}(x,t) = \nabla \cdot (D(x,t,u)\nabla u(x,t)), \quad \text{em } Q \times [0,T] \tag{2.0.1}$$

onde a solução $u(x,t)$ representa diferentes fases da imagem filtrada, x é a coordenada espacial definida no quadrado $Q = [1,N_1] \times [1,N_2]$ e t é o tempo de coordenadas definido no intervalo de $[0,T]$ onde T representa o tempo de difusão. O coeficiente de difusão D tem de ser devidamente definido a fim de evitar a difusão entre as transições rápidas de intensidade na imagem e, por conseguinte, um efeito de desfocagem. Perona e Malik [Perona and Malik, 1990] propuseram a utilização de um coeficiente de difusão baseada no gradiente de imagem, ver equação 1.1.2, de modo a distinguir entre os rebordos e as regiões constantes. No entanto, no passo inicial da imagem, onde o nível de ruído é alto, o gradiente é instável. Para ultrapassar este problema, Gilboa [Gilboa et al., 2004] sugeriu considerar um filtro complexo apropriado

[1] Objetiva restaurar as partes em falta ou danificadas de uma imagem, a fim de torná-la mais legível.

do formulário

$$D = \frac{e^{i\vartheta}}{1 + \left(\frac{Im(u)}{\kappa\vartheta}\right)^2}. \tag{2.0.2}$$

onde $\vartheta \approx 0$ e κ é um coeficiente positivo. Foi comprovado que o filtro é eficiente, desde que

$$\lim_{\vartheta \to 0} \frac{Im(u(.,t))}{t\vartheta} = G * \Delta I_0$$

onde I_0 é a imagem inicial, G é o gaussiano e $*$ representa o operador de convolução.

Nesse caso o D complexo imita o Laplaciano, e assim tem-se um coeficiente D, que dificulta a difusão sobre as alterações bruscas de intensidade na imagem, sem o cálculo de derivadas.

Os métodos numéricos de diferenças finitas são baseados nas propriedades da série de Taylor, permitindo obter aproximações das suas derivadas, utilizando nós igualmente espaçados com espaçamento h. Em [Araújo et al., 2012] um rigoroso resultado de estabilidade foi provado para uma família de esquemas de diferenças finitas para (2.0.1) com coeficiente de difusão complexo. O resultado de convergência também foi alcançado em [Araújo et al., 2013].

Embora os métodos de diferenças finitas implícito e semi-implícito sejam incondicionalmente estáveis, os investigadores tendem a implementar e usar esquemas explícitos para o processo de difusão. O esquema explícito é mais fácil de ser implementado, mas o número de iterações é limitado anteriormente pela condição de estabilidade, o que leva à necessidade de um maior número de iterações. Por outro lado, o método semi-implícito tem a necessidade de resolver um sistema linear a cada passo, embora o número de passos possa ser consideravelmente menor. No caso do método implícito, a cada passo é necessário resolver um sistema de equações não lineares, que após a linearização se traduzem na resolução iterativa de sistemas lineares para aproximar a solução do sistema não linear inicial.

Desta forma a escolha entre um método explícito, semi-implícito ou implícito tem necessariamente a ver com o equilíbrio entre o número de iterações necessário e o custo computacional de cada iteração para cada esquema considerado, de forma a garantir globalmente uma boa performance de eliminação de ruído com baixo custo computacional.

Seja I_0 a imagem original (com ruído) de tamanho $N_1 \times N_2$. A difusão complexa é usualmente

baseada na solução numérica da equação diferencial parcial (2.0.1). Para ter um problema bem colocado, a equação (2.0.1) tem que ser completada por uma condição inicial na forma

$$u(x,0) = I_0, \tag{2.0.3}$$

Além disso, são necessárias condições de fronteira definida na fronteira Γ do conjunto Q. Serão consideradas condições de fronteira de Dirichlet

$$u(x,t) = g(x,t), \quad x \in \Gamma, t \in [0,T], \tag{2.0.4}$$

o que significa que os limites da imagem são mantidos fixos com algum g dado, ou, em alternativa, serão consideradas as condições de fronteira de Neumann

$$\frac{\partial u}{\partial \nu}(x,t) = 0, \quad x \in \Gamma, t \in [0,T], \tag{2.0.5}$$

que significa que não há o fluxo de intensidade no sentido normal, onde ν é o vetor normal unitário exterior.

2.1 Discretização

Considera-se uma malha igualmente espaçada em Q, com espaçamento espacial $h_1 = h_2 = 1$ na primeira e na segunda direção de coordenada. A malha está, por conseguinte, definida pelo conjunto de pontos

$$x_{j,k} = (j,k), \quad j = 0,1,2,\ldots,N_1+1, k = 0,1,2,\ldots,N_2+1,$$

em que os pontos com coordenadas $j = 0$, $j = N_1 + 1$, $k = 0$ ou $k = N_2 + 1$ correspondem aos pontos onde são impostas as condições numa fronteira artificial. Além disso, considera-se o espaçamento do tempo h_t, que define o conjunto de pontos

$$t_0 = 0, t_{m+1} = t_m + h_t, \quad m = 0,1,\ldots,N_t.$$

tal que $t_{N_t} = T$. Portanto, pode-se definir uma malha Q_h definida pelo conjunto de pontos

$$(x_{j,k}, t_m), \quad j = 0,1,2,\ldots,N_1+1, k = 0,1,2,\ldots,N_2+1, m = 0,1,\ldots,N_t.$$

A aproximação de primeira ordem para a primeira derivada por diferenças finitas progressiva em t é dada por

$$y'(t) \approx \frac{y(t+h)-y(t)}{h} \tag{2.1.6}$$

enquanto que a aproximação de segunda ordem para a segunda derivada centrada em t é dada por

$$y''(t) \approx \frac{y(t+h)-2y(t)+y(t-h)}{h^2} \tag{2.1.7}$$

Considera-se agora o método geral de diferenças finitas (ver [Araújo et al., 2012]) para a equação diferencial (2.0.1) originado por considerar, de modo genérico, a diferença finita de primeira ordem no tempo, e as diferenças finitas de segunda ordem no espaço, que é dada por

$$\begin{aligned} U_{j,k}^{m+1} = U_{j,k}^{m} &+ \frac{h_t}{2h_1^2}\left[\left(D_{(j+1,k)}^{m,\theta,\mu} + D_{(j,k)}^{m,\theta,\mu}\right)U_{(j+1,k)}^{m+\theta} + \left(D_{(j,k)}^{m,\theta,\mu} + D_{(j-1,k)}^{m,\theta,\mu}\right)U_{(j-1,k)}^{m+\theta}\right. \\ &\left. - \left(D_{(j+1,k)}^{m,\theta,\mu} + 2D_{(j,k)}^{m,\theta,\mu} + D_{(j-1,k)}^{m,\theta,\mu}\right)U_{(j,k)}^{m+\theta}\right] \\ &+ \frac{h_t}{2h_2^2}\left[\left(D_{(j,k+1)}^{m,\theta,\mu} + D_{(j,k)}^{m,\theta,\mu}\right)U_{(j,k+1)}^{m+\theta} + \left(D_{(j,k)}^{m,\theta,\mu} + D_{(j,k-1)}^{m,\theta,\mu}\right)U_{(j,k-1)}^{m+\theta}\right. \\ &\left. - \left(D_{(j,k+1)}^{m,\theta,\mu} + 2D_{(j,k)}^{m,\theta,\mu} + D_{(j,k-1)}^{m,\theta,\mu}\right)U_{(j,k)}^{m+\theta}\right] \end{aligned} \tag{2.1.8}$$

para $j = 1,2,\ldots,N_1$, $k = 1,2,\ldots,N_2$, $m = 0,1,\ldots,N_t-1$ onde $U_{j,k}^m$ é a aproximação da solução $u(x_{j,k}, t_m)$,

$$D_{(j,k)}^{m,\theta,\mu} = D\left(x_{j,k}, t^{m+\theta}, U_{j,k}^{m+\theta\mu}\right),$$

e $\mu \in \{0,1\}$, $t^{m+\theta} = \theta t^{m+1} + (1-\theta)t^m$, $\theta \in [0,1]$ e

$$V_{j,k}^{m+\theta} = \theta V_{j,k}^{m+1} + (1-\theta)V_{j,k}^{m}, \quad V = U, D,$$

Diversas escolhas (θ, μ) traduzem-se em diversos esquemas numéricos diferentes.

2.2 Método Explícito

O método explícito é dado por $\theta = 0$ e $\mu = 0$, ou seja:

$$\begin{aligned} U_{j,k}^{m+1} = & \left[1 - \frac{h_t}{2h_1^2}\left(D_{(j+1,k)}^m + 2D_{(j,k)}^m + D_{(j-1,k)}^m\right)\right. \\ & \left. - \frac{h_t}{2h_2^2}\left(D_{(j,k+1)}^m + 2D_{(j,k)}^m + D_{(j,k-1)}^m\right)\right] U_{j,k}^m \\ & + \frac{h_t}{2h_1^2}\left[\left(D_{(j+1,k)}^m + D_{(j,k)}^m\right) U_{(j+1,k)}^m + \left(D_{(j,k)}^m + D_{(j-1,k)}^m\right) U_{(j-1,k)}^m\right] \\ & + \frac{h_t}{2h_2^2}\left[\left(D_{(j,k+1)}^m + D_{(j,k)}^m\right) U_{(j,k+1)}^m + \left(D_{(j,k)}^m + D_{(j,k-1)}^m\right) U_{(j,k-1)}^m\right] \end{aligned} \tag{2.2.9}$$

para $j = 1, 2, \ldots, N_1$, $k = 1, 2, \ldots, N_2$, $m = 0, 1, \ldots, N_t - 1$. Este método é conhecido por ser estável (ver [Araújo et al., 2012]) se

$$h_t \leq \frac{\min(h_1, h_2)}{4 \max \frac{\left|D_{j,k}^m\right|^2}{\mathrm{Re}\left(D_{j,k}^m\right)}}. \tag{2.2.10}$$

É importante salientar que para o método explícito não é necessário resolver um sistema linear. Para obter o valor $U_{j,k}^{m+1}$ na nova iteração $m+1$, somente os valores da iteração anterior m são usados. Como não se abordou numericamente o método explícito neste trabalho, deixa-se a análise das condições inicial e de fronteira para os esquemas seguintes.

2.3 Método Semi-implícito

O método semi-implícito é definido por $\theta = 1$ e $\mu = 0$. Este método é conhecido por ser incondicionalmente estável (ver [Araújo et al., 2013]), ou seja

$$\begin{aligned}
&\left[1+\frac{h_t}{2h_1^2}\left(D^m_{(j+1,k)}+2D^m_{(j,k)}+D^m_{(j-1,k)}\right)\right.\\
&\qquad\left.+\frac{h_t}{2h_2^2}\left(D^m_{(j,k+1)}+2D^m_{(j,k)}+D^m_{(j,k-1)}\right)\right]U^{m+1}_{(j,k)}\\
&-\frac{h_t}{2h_1^2}\left[\left(D^m_{(j+1,k)}+D^m_{(j,k)}\right)U^{m+1}_{(j+1,k)}+\left(D^m_{(j,k)}+D^m_{(j-1,k)}\right)U^{m+1}_{(j-1,k)}\right]\\
&-\frac{h_t}{2h_2^2}\left[\left(D^m_{(j,k+1)}+D^m_{(j,k)}\right)U^{m+1}_{(j,k+1)}+\left(D^m_{(j,k)}+D^m_{(j,k-1)}\right)U^{m+1}_{(j,k-1)}\right]\\
&=U^m_{(j,k)} \qquad (2.3.11)
\end{aligned}$$

para $j = 1,2,\ldots,N_1$, $k = 1,2,\ldots,N_2$, $m = 0,1,\ldots,N_t-1$. Para este método numérico, um sistema linear deve ser resolvido em cada iteração.

Note que para um m fixo, existem $N_1 \times N_2$ equações. No entanto existem $N_1 \times N_2 + 2(N_1+N_2)$ incógnitas, pois para $j = 1,\ldots,N_1$ e $k = 1,\ldots,N_2$, os termos $(j \pm 1, k)$ e $(j, k \pm 1)$ são considerados. Assim, de forma a se ter um sistema bem definido são necessárias mais $2(N_1+N_2)$ equações para acomodar os termos fronteira, ou, dito de outra forma, de eliminar $2(N_1+N_2)$ variáveis. As condições de fronteira que permitirão atingir este objetivo, de forma a ter um problema bem posto, serão abordadas a seguir.

Note que como cada equação $U^{m+1}_{j,k}$ depende de 4 (desconhecidos) valores $U^{m+1}_{j\pm1,k\pm1}$ e de valores conhecidos (da iterada anterior) $U^m_{j,k}$, $D^m_{j,k}$ e $D^m_{j\pm1,k\pm1}$, um sistema de equações surge de (2.3.11).

A condição inicial, dada por:

$$U^0_{j,k} = I_0(j,k), \quad j = 1,2,\ldots,N_1,\ k = 1,2,\ldots,N_2. \qquad (2.3.12)$$

estabelece o ponto de partida para calcular a nova iteração, a partir da anterior, para $U^0_{j,k}, j = 1,2,...N_1, k = 1,2,...,N_2$. A condição inicial é independente da escolha da condição de fronteira.

De entre as condições de fronteiras existentes trabalhou-se especificamente com Dirichlet e Neumann.

2.3.1 Condição de Fronteira de Dirichlet

No caso da condição de Dirichlet, assumiu-se que na fronteira (artificial) o valor da imagem não se altera, ou seja, tem-se formalmente:

$$\begin{aligned} U^m_{j,0} = I_0(j,1), U^m_{j,N_2+1} = I_0(j,N_1), \quad j = 1,2,\ldots,N_1, \\ U^m_{0,k} = I_0(1,k), U^m_{N_1+1,k} = I_0(N_1,k), \quad k = 1,2,\ldots,N_2, \end{aligned} \tag{2.3.13}$$

para $m = 0,1,\ldots,N_t$.

Como exemplo, tomando a equação (2.3.11) para $j = k = 1$ e considerando as condições de Dirichlet, obtêm-se:

$$\begin{aligned} &\left[1 + \frac{h_t}{2h_1^2}\left(D^m_{(2,1)} + 2D^m_{(1,1)} + D^m_{(0,1)}\right) + \frac{h_t}{2h_2^2}\left(D^m_{(1,2)} + 2D^m_{(1,1)} + D^m_{(1,0)}\right)\right] U^{m+1}_{(1,1)} \\ &- \frac{h_t}{2h_1^2}\left(D^m_{(2,1)} + D^m_{(1,1)}\right) U^{m+1}_{(2,1)} - \frac{h_t}{2h_2^2}\left(D^m_{(1,2)} + D^m_{(1,1)}\right) U^{m+1}_{(1,2)} \\ &= U^m_{(1,1)} + \frac{h_t}{2h_1^2}\left(D^m_{(1,1)} + D^m_{(0,1)}\right) I_0(1,1) + \frac{h_t}{2h_2^2}\left(D^m_{(1,1)} + D^m_{(1,0)}\right) I_0(1,1) \end{aligned} \tag{2.3.14}$$

Pode-se facilmente observar que os termos de $U_{(0,k)}$ e $U_{(j,0)}$ desaparecem. Para $D_{(0,k)}$, $D_{(j,0)}$ considera-se os valores de D para o vizinho interior da grelha mais próximo, ou seja, $D_{(1,k)}$, $D_{(j,1)}$, respectivamente.

2.3.2 Condição de Fronteira de Neumann

A condição de fronteira de Neumann, quando aplicada a uma equação diferencial ordinária ou parcial, especifica os valores que a derivada normal de uma solução possui na fronteira do domínio espacial. Enquanto a condição de fronteira de Dirichlet especifica o valor da função no contorno, a condição de fronteira de Neumann especifica a derivada normal da função no domínio, ou seja, fixa o valor do fluxo. Em particular, a condição de Neumann homogênea (que diz que a derivada normal é nula) indica que o fluxo é nulo na fronteira espacial do domínio.

Formalmente, a condição de Neumann é definida pela igualdade (aproximação de segunda

ordem por diferenças finitas):

$$\begin{aligned} U^m_{j,0} = U^m_{j,2}, U^m_{j,N_2+1} = U^m_{j,N_2-1}, \quad j = 1,2,\ldots,N_1, \\ U^m_{0,k} = U^m_{2,k}, U^m_{N_1+1,k} = U^m_{N_1-1,k}, \quad k = 1,2,\ldots,N_2, \end{aligned} \tag{2.3.15}$$

Da mesma forma que para o caso de Dirichlet, tem-se como exemplo a equação geral, para $j = k = 1$ e considerando as condições de Neumann, tem-se:

$$\begin{aligned} &\left[1 + \frac{h_t}{2h_1^2}\left(D^m_{(2,1)} + 2D^m_{(1,1)} + D^m_{(0,1)}\right) + \frac{h_t}{2h_2^2}\left(D^m_{(1,2)} + 2D^m_{(1,1)} + D^m_{(1,0)}\right)\right] U^{m+1}_{(1,1)} \\ &- \frac{h_t}{2h_1^2}\left(D^m_{(2,1)} + 2D^m_{(1,1)} + D^m_{(0,1)}\right) U^{m+1}_{(2,1)} \\ &- \frac{h_t}{2h_2^2}\left(D^m_{(1,2)} + 2D^m_{(1,1)} + D^m_{(1,0)}\right) U^{m+1}_{(1,2)} = U^m_{(1,1)} \end{aligned} \tag{2.3.16}$$

Assim como no caso de Dirichlet, em que os termos $U_{(0,k)}$ e $U_{(j,0)}$ também desaparecem.

2.3.3 Procedimento do Algoritmo Semi-implícito

Foi criado um algoritmo em Octave[2] (versão 3.6.1) para criar e resolver esse sistema linear. A entrada inicial da matriz consiste em uma imagem U, que quando lida pelo Octave é representada por uma matriz de pixels. Essa imagem corresponde à condição inicial e em termos práticos é uma imagem cujo ruído se quer reduzir ou mesmo eliminar. Visto que para o método semi-implícito para o cálculo de $U^{(m+1)}$ apenas são considerados os valores de D da iterada anterior (D^m), o primeiro passo foi calcular o valor de D para todas coordenadas (j,k) da iterada anterior.

A partir dos valores de D devidamente armazenados em matrizes, criou-se a matriz e o vetor do segundo membro do sistema linear, chamados de A e B, respectivamente. Para facilitar a implementação, dividiu-se esse problema em 5 partes, uma para alimentar a diagonal principal da Matriz A, e outras quatro para alimentar outras 4 diagonais diferentes de zero na matriz, como é característico de matrizes para métodos de diferenças finitas (que utilizam os pontos $(j \pm 1, k)$ e $(j, k \pm 1)$) como os usados.

[2]GNU Octave é uma linguagem computacional, desenvolvida para computação matemática.

Ao final dessas 5 etapas, tem-se a Matriz A e o Vetor B devidamente alimentados. Portanto, resolve-se o sistema linear com o auxílio do Octave e tem-se uma nova matriz (nova imagem), que corresponde à solução no próximo passo de tempo (que já terá menos ruído), e que será considerada como entrada para o cálculo do passo seguinte do método.

Definido o tempo final T de difusão e o espaçamento no tempo h_t conforme o caso e a aplicação em vista, o número de passos é definido pelo quociente de T por h_t.

2.4 Método Implícito

A discretização (2.3.11) correspondente ao método semi-implícito conduz a um sistema linear. Assim, no passo de tempo t_m, pretende-se determinar a solução $U^{m+1}_{(j,k)}$ (para $j = 1, 2, \ldots, N_1$, $k = 1, 2, \ldots, N_2$,) no instante t_{m+1}, conhecendo a solução $U^m_{(j,k)}$ no instante anterior e usando-a para o cálculo de $D^m_{(j,k)}$. Note que neste caso, todos os coeficientes dos termos $U^{m+1}_{(.,.)}$ são conhecidos, pelo que se pode estabelecer um sistema linear.

No caso de método (totalmente) implícito, considera-se (2.1.8) com $\mu = \theta = 1$, o que resulta em

$$\begin{aligned}
&\left[1 + \frac{h_t}{2h_1^2}\left(D^{m+1}_{(j+1,k)} + 2D^{m+1}_{(j,k)} + D^{m+1}_{(j-1,k)}\right)\right. \\
&\qquad \left. + \frac{h_t}{2h_2^2}\left(D^{m+1}_{(j,k+1)} + 2D^{m+1}_{(j,k)} + D^{m+1}_{(j,k-1)}\right)\right] U^{m+1}_{(j,k)} \\
&- \frac{h_t}{2h_1^2}\left[\left(D^{m+1}_{(j+1,k)} + D^{m+1}_{(j,k)}\right) U^{m+1}_{(j+1,k)} + \left(D^{m+1}_{(j,k)} + D^{m+1}_{(j-1,k)}\right) U^{m+1}_{(j-1,k)}\right] \\
&- \frac{h_t}{2h_2^2}\left[\left(D^{m+1}_{(j,k+1)} + D^{m+1}_{(j,k)}\right) U^{m+1}_{(j,k+1)} + \left(D^{m+1}_{(j,k)} + D^{m+1}_{(j,k-1)}\right) U^{m+1}_{(j,k-1)}\right] \\
&= U^m_{(j,k)}
\end{aligned} \tag{2.4.17}$$

em que

$$D^{m+1}_{(j,k)} = \frac{e^{i\vartheta}}{1 + \left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}, \tag{2.4.18}$$

em que a parte imaginária de $U^m_{(j,k)}$ é definida por $U^m_{I(j,k)}$. Note que neste caso, a equação (2.4.17) não é linear nos termos $U^{m+1}_{(j,k)}$. Pode-se reescrever (2.4.17) em função da parte real $U^{m+1}_{R(j,k)}$ e

imaginária $U^{m+1}_{R(j,k)}$ de $U^{m+1}_{(j,k)}$ notando que

$$U^{m+1}_{(j,k)} = U^{m+1}_{R(j,k)} + iU^{m+1}_{I(j,k)},$$

e sem depender de $D^{m+1}_{(j,k)}$, mas sim substituindo pela sua expressão (2.4.18). Tem-se então a equação (2.4.17) na forma

$$\begin{aligned}
&F_{R(j,k)}\left(U^{m+1}_{R(j,k)}, U^{m+1}_{I(j,k)}, U^{m+1}_{R(j+1,k)}, U^{m+1}_{I(j+1,k)},\right.\\
&\qquad \left. U^{m+1}_{R(j-1,k)}, U^{m+1}_{I(j-1,k)}, U^{m+1}_{R(j,k+1)}, U^{m+1}_{I(j,k+1)}, U^{m+1}_{R(j,k-1)}, U^{m+1}_{I(j,k-1)}\right) = 0\\
&F_{I(j,k)}\left(U^{m+1}_{R(j,k)}, U^{m+1}_{I(j,k)}, U^{m+1}_{R(j+1,k)}, U^{m+1}_{I(j+1,k)},\right.\\
&\qquad \left. U^{m+1}_{R(j-1,k)}, U^{m+1}_{I(j-1,k)}, U^{m+1}_{R(j,k+1)}, U^{m+1}_{I(j,k+1)}, U^{m+1}_{R(j,k-1)}, U^{m+1}_{I(j,k-1)}\right) = 0
\end{aligned}$$

em que $F_{R(j,k)}$ é a parte real da equação e $F_{I(j,k)}$ a sua parte imaginária, ou seja, como $e^{i\vartheta} = \cos(\vartheta) + i\sin(\vartheta)$ tem-se a parte real

$$\begin{aligned}
F_{R(j,k)} = &\left[1 + \frac{h_t}{2h_1^2}\left(\frac{\cos(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{2\cos(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{\cos(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right)\right.\\
&\left. + \frac{h_t}{2h_2^2}\left(\frac{\cos(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}} + \frac{2\cos(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{\cos(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right)\right] U^{m+1}_{R(j,k)}\\
&- \left[\frac{h_t}{2h_1^2}\left(\frac{\sin(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{2\sin(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{\sin(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right)\right.\\
&\left. + \frac{h_t}{2h_2^2}\left(\frac{\sin(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}} + \frac{2\sin(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{\sin(\vartheta)}{1 + \frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right)\right] U^{m+1}_{I(j,k)}
\end{aligned} \tag{2.4.19}$$

$$
\begin{aligned}
&-\frac{h_t}{2h_1^2}\Bigg[\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{R(j+1,k)}\\
&\qquad-\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{I(j+1,k)}\\
&\qquad+\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{R(j-1,k)}\\
&\qquad-\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{I(j-1,k)}\Bigg]\\
&-\frac{h_t}{2h_2^2}\Bigg[\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{R(j,k+1)}\\
&\qquad-\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{I(j,k+1)}\\
&\qquad+\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{R(j,k-1)}\\
&\qquad-\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{I(j,k-1)}\Bigg]-U^{m}_{R(j,k)},
\end{aligned}
$$

e a parte imaginária

$$
\begin{aligned}
F_{I(j,k)} = &\left[1+\frac{h_t}{2h_1^2}\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{2\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right)\right.\\
&\left.+\frac{h_t}{2h_2^2}\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}}+\frac{2\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right)\right]U^{m+1}_{I(j,k)}\\
&+\left[\frac{h_t}{2h_1^2}\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{2\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right)\right.\\
&\left.+\frac{h_t}{2h_2^2}\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}}+\frac{2\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right)\right]U^{m+1}_{R(j,k)}\\
&-\frac{h_t}{2h_1^2}\left[\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{I(j+1,k)}\right.\\
&\quad+\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{R(j+1,k)}\\
&\quad+\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{I(j-1,k)}\\
&\quad\left.+\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{R(j-1,k)}\right]\\
&-\frac{h_t}{2h_2^2}\left[\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{I(j,k+1)}\right.\\
&\quad\left.+\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{R(j,k+1)}\right.
\end{aligned}
$$

$$+\left(\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{I(j,k-1)}$$
$$+\left(\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right)U^{m+1}_{R(j,k-1)}\Bigg]-U^{m}_{I(j,k)}.$$

Note que a equação (2.4.17) é em torno no ponto (j,k), ou seja, o sistema completo de equações não lineares é composto por $N_1 \times N_2$. Ao desdobrar a equação nas partes real e imaginária, passa-se a ter $2 \times N_1 \times N_2$ equações. Pode-se agora reescrever esse sistema de equações como

$$F(U^{m+1}) = 0$$

em que

$$F(U) = \begin{bmatrix} F_{R(1,1)} \\ F_{R(1,2)} \\ \vdots \\ F_{R(i,j)} \\ \vdots \\ F_{R(N_1,N_2)} \\ F_{I(1,1)} \\ F_{I(1,2)} \\ \vdots \\ F_{I(i,j)} \\ \vdots \\ F_{I(N_1,N_2)} \end{bmatrix}.$$

2.5 Método de Newton

Para aproximar a solução do sistema de equações, utilizar-se-á o método de Newton. Assim, a cada passo do método implícito considera-se a iterada inicial dada pela solução no instante de

tempo anterior, ou seja,

$$x_0 = [U^m_{R(1,1)}, U^m_{R(1,2)}, \ldots, U^m_{R(N_1,N_2)}, U^m_{I(1,1)}, U^m_{I(1,2)}, \ldots, U^m_{I(N_1,N_2)}]$$

Determinou-se depois a derivada F' de F, que neste caso é uma matriz $2 \times N_1 \times N_2$ dada pelo jacobiano de F com entradas

$$F' = \left[\begin{array}{ccc|ccc} & \vdots & & & \vdots & \\ \cdots & \dfrac{\partial F_{R(j,k)}}{\partial U^{m+1}_{R(n,p)}} & \cdots & \cdots & \dfrac{\partial F_{R(j,k)}}{\partial U^{m+1}_{I(n,p)}} & \cdots \\ & \vdots & & & \vdots & \\ \hline & \vdots & & & \vdots & \\ \cdots & \dfrac{\partial F_{I(j,k)}}{\partial U^{m+1}_{R(n,p)}} & \cdots & \cdots & \dfrac{\partial F_{I(j,k)}}{\partial U^{m+1}_{I(n,p)}} & \cdots \\ & \vdots & & & \vdots & \end{array}\right]$$

em que cada um dos blocos é uma matriz $N_1 \times N_2$ dada por

$$\frac{\partial F_{(j,k)}}{\partial U^{m+1}_{(n,p)}} = \left[\begin{array}{cccc} \dfrac{\partial F_{(1,1)}}{\partial U^{m+1}_{(1,1)}} & \dfrac{\partial F_{(1,1)}}{\partial U^{m+1}_{(1,2)}} & \cdots & \dfrac{\partial F_{(1,1)}}{\partial U^{m+1}_{(N_1,N_2)}} \\ \dfrac{\partial F_{(1,2)}}{\partial U^{m+1}_{(1,1)}} & \dfrac{\partial F_{(1,2)}}{\partial U^{m+1}_{(1,2)}} & \cdots & \dfrac{\partial F_{(1,2)}}{\partial U^{m+1}_{(N_1,N_2)}} \\ \vdots & & \ddots & \vdots \\ \dfrac{\partial F_{(N_1,N_2)}}{\partial U^{m+1}_{(1,1)}} & \dfrac{\partial F_{(N_1,N_2)}}{\partial U^{m+1}_{(1,2)}} & \cdots & \dfrac{\partial F_{(N_1,N_2)}}{\partial U^{m+1}_{(N_1,N_2)}} \end{array}\right]$$

Assim, para obter a primeira iterada x_1, resolveu-se em ordem a h, o sistema linear

$$F'(x_0)h = F(x_0)$$

e atualiza-se $x_1 = x_0 - h$. Repete-se o processo

$$\begin{cases} F'(x_n)h = F(x_n), \\ x_{n+1} = x_n - h \end{cases}$$

um número J de iterações suficiente de forma a termos a aproximação

$$U^{m+1}_{(.,.)} \approx x_J.$$

Como critério de paragem para o método de Newton, definiu-se que a norma L^2 discreta da diferença entre iteradas consecutivas é menor que uma tolerância pré-definida. Assim, tem-se a aproximação no instante de tempo t_{m+1} e pode-se repetir o processo até o tempo de difusão T predefinido.

Sendo que para o método de Newton são necessárias todas as derivadas, da parte real e da parte imaginária da função F, em função de todas partes reais e imaginárias de $U_{(.,.)}^{m+1}$. Estas derivadas são apresentadas nas seções 2.4.2 e 2.4.3.

2.5.1 Procedimento do Algoritmo Implícito

Tem-se o seguinte pseudocódigo para o método implícito:

U^0 `(Iterada inicial)`
`for` $n = 0, 1, \ldots, N$ `(passos do método implícito)`
  $x_0 \leftarrow U^n$ `(iterada inicial do método de Newton)`
  `while` $j \leq$ `MaxIter e` $L2 \leq L2Max$ `(iteradas Met. Newton)`
    `Criar o vetor` $F(x_j, U^n)$
    `Criar a matriz` $F'(x_j)$
    `Resolver o sistema` $F'(x_j)h = F(x_j, U^n)$
    `Nova iterada` $x_{j+1} = x_j - h$
  `end`
  $U^{n+1} \leftarrow x_J$ `(Novo passo do met. Implícito)`
`end`
U^N `(Aproximação final, imagem sem ruído)`

2.5.2 Derivadas da Parte Real da Função F

A seguir são listadas as derivadas da função F da parte real da imagem, em relação à parte real e em relação à parte imaginária.

Tem-se assim a derivada para alimentar a diagonal do respectivo bloco:

$$
\begin{aligned}
\frac{\partial F_{R(j,k)}}{\partial U^{m+1}_{I(j,k)}} = &\left\{ \frac{h_t}{h_1^2}\left[-\frac{2\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} \right] + \frac{h_t}{h_2^2}\left[-\frac{2\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} \right] \right\} U^{m+1}_{R(j,k)} \\
&- \left\{ \frac{h_t}{h_1^2}\left[-\frac{2\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} \right] + \frac{h_t}{h_2^2}\left[-\frac{2\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} \right] \right\} U^{m+1}_{I(j,k)} \\
&- \left\{ \frac{h_t}{2h_1^2}\left[\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{2\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}} \right] \right. \\
&\left. + \frac{h_t}{2h_2^2}\left[\frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}} + \frac{2\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}} + \frac{\sin(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}} \right] \right\} \\
&- \frac{h_t}{h_1^2}\left\{ -\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} U^{m+1}_{R(j+1,k)} + \frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} U^{m+1}_{I(j+1,k)} \right. \\
&\left. - \frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} U^{m+1}_{R(j-1,k)} + \frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} U^{m+1}_{I(j-1,k)} \right\} \\
&- \frac{h_t}{h_2^2}\left\{ -\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} U^{m+1}_{R(j,k+1)} + \frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} U^{m+1}_{I(j,k+1)} \right. \\
&\left. - \frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} U^{m+1}_{R(j,k-1)} + \frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2} U^{m+1}_{I(j,k-1)} \right\},
\end{aligned}
$$

e as restantes derivadas de interesse para alimentar as restantes diagonais do bloco:

$$\frac{\partial F_{R(j,k)}}{\partial U^{m+1}_{I(j+1,k)}} = -\frac{h_t}{h_1^2}\left[\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j+1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j+1,k)}\right)^2\right)^2}\right]U^{m+1}_{R(j,k)} + \frac{h_t}{h_1^2}\left[\frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j+1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j+1,k)}\right)^2\right)^2}\right]U^{m+1}_{I(j,k)}$$
$$-\frac{h_t}{h_1^2}\left[-\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j+1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j+1,k)}\right)^2\right)^2}U^{m+1}_{R(j+1,k)} + \frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j+1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j+1,k)}\right)^2\right)^2}U^{m+1}_{I(j+1,k)}\right.$$
$$\left.-\frac{1}{2}\left(\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j+1,k)}}{\kappa\vartheta}\right)^2}+\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}\right)\right],$$

$$\frac{\partial F_{R(j,k)}}{\partial U^{m+1}_{I(j-1,k)}} = -\frac{h_t}{h_1^2}\left[\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j-1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j-1,k)}\right)^2\right)^2}\right]U^{m+1}_{R(j,k)} + \frac{h_t}{h_1^2}\left[\frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j-1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j-1,k)}\right)^2\right)^2}\right]U^{m+1}_{I(j,k)}$$
$$-\frac{h_t}{h_1^2}\left[-\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j-1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j-1,k)}\right)^2\right)^2}U^{m+1}_{R(j-1,k)} + \frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j-1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j-1,k)}\right)^2\right)^2}U^{m+1}_{I(j-1,k)}\right.$$
$$\left.-\frac{1}{2}\left(\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}+\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j-1,k)}}{\kappa\vartheta}\right)^2}\right)\right],$$

$$\frac{\partial F_{R(j,k)}}{U^{m+1}_{I(j,k+1)}} = -\frac{h_t}{h_2^2}\left[\frac{\cos(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j,k+1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k+1)}\right)^2\right)^2}\right]U^{m+1}_{R(j,k)} + \frac{h_t}{h_2^2}\left[\frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k+1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k+1)}\right)^2\right)^2}\right]U^{m+1}_{I(j,k)}$$
$$-\frac{h_t}{h_2^2}\left[-\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k+1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k+1)}\right)^2\right)^2}U^{m+1}_{R(j,k+1)} + \frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k+1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k+1)}\right)^2\right)^2}U^{m+1}_{I(j,k+1)}\right.$$
$$\left.-\frac{1}{2}\left(\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}+\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k+1)}}{\kappa\vartheta}\right)^2}\right)\right],$$

$$\frac{\partial F_{R(j,k)}}{U^{m+1}_{I(j,k-1)}} = -\frac{h_t}{h_2^2}\left[\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k-1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k-1)}\right)^2\right)^2}\right]U^{m+1}_{R(j,k)} + \frac{h_t}{h_2^2}\left[\frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k-1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k-1)}\right)^2\right)^2}\right]U^{m+1}_{I(j,k)}$$
$$-\frac{h_t}{h_2^2}\left[-\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k-1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k-1)}\right)^2\right)^2}U^{m+1}_{R(j,k-1)} + \frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k-1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k-1)}\right)^2\right)^2}U^{m+1}_{I(j,k-1)}\right.$$
$$\left.-\frac{1}{2}\left(\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}+\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k-1)}}{\kappa\vartheta}\right)^2}\right)\right],$$

e as derivadas da parte real da função F_R em relação à parte real da imagem U_R

$$\frac{\partial F_{R(j,k)}}{U^{m+1}_{R(j,k)}} = 1+\frac{h_t}{2h_1^2}\left[\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j+1,k)}}{\kappa\vartheta}\right)^2}+\frac{2\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}+\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j-1,k)}}{\kappa\vartheta}\right)^2}\right]$$
$$+\frac{h_t}{2h_2^2}\left[\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k+1)}}{\kappa\vartheta}\right)^2}+\frac{2\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}+\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k-1)}}{\kappa\vartheta}\right)^2}\right],$$

$$\frac{\partial F_{R(j,k)}}{U^{m+1}_{R(j+1,k)}} = -\frac{h_t}{2h_1^2}\left[\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j+1,k)}}{\kappa\vartheta}\right)^2}+\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}\right],$$

$$\frac{\partial F_{R(j,k)}}{U^{m+1}_{R(j-1,k)}} = -\frac{h_t}{2h_1^2}\left[\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j-1,k)}}{\kappa\vartheta}\right)^2}+\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}\right],$$

$$\frac{\partial F_{R(j,k)}}{U^{m+1}_{R(j,k+1)}} = -\frac{h_t}{2h_2^2}\left[\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k+1)}}{\kappa\vartheta}\right)^2}+\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}\right],$$

$$\frac{\partial F_{R(j,k)}}{U^{m+1}_{R(j,k-1)}} = -\frac{h_t}{2h_2^2}\left[\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k-1)}}{\kappa\vartheta}\right)^2}+\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}\right].$$

2.5.3 Derivadas da Parte Imaginária da Função F

A seguir são listadas as derivadas da função F da parte imaginária da imagem, em relação à parte real e em relação à parte imaginária.

Tem-se assim a derivada para alimentar a diagonal do respectivo bloco:

$$\begin{aligned}\frac{\partial F_{I(j,k)}}{\partial U^{m+1}_{I(j,k)}} = &\left\{\frac{h_t}{h_1^2}\left[-\frac{2\cos(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}\right]+\frac{h_t}{h_2^2}\left[-\frac{2\cos(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}\right]\right\}U^{m+1}_{I(j,k)}\\
&+\left\{1+\frac{h_t}{2h_1^2}\left[\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j+1,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{2\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j-1,k)}\right)^2}{\kappa^2\vartheta^2}}\right]\right.\\
&\left.+\frac{h_t}{2h_2^2}\left[\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k+1)}\right)^2}{\kappa^2\vartheta^2}}+\frac{2\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k)}\right)^2}{\kappa^2\vartheta^2}}+\frac{\cos(\vartheta)}{1+\frac{\left(U^{m+1}_{I(j,k-1)}\right)^2}{\kappa^2\vartheta^2}}\right]\right\}\end{aligned}$$

$$
\begin{aligned}
&+\left\{\frac{h_t}{h_1^2}\left[-\frac{2\sin(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}\right]+\frac{h_t}{h_2^2}\left[-\frac{2\sin(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}\right]\right\}U^{m+1}_{R(j,k)}\\
&+\frac{h_t}{h_1^2}\left\{\left[\frac{\cos(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}U^{m+1}_{I(j+1,k)}+\frac{\sin(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}U^{m+1}_{R(j+1,k)}\right]\right.\\
&\left.+\left[\frac{\cos(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}U^{m+1}_{I(j-1,k)}+\frac{\sin(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}U^{m+1}_{R(j-1,k)}\right]\right\}\\
&+\frac{h_t}{h_2^2}\left\{\left[\frac{\cos(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}U^{m+1}_{I(j,k+1)}+\frac{\sin(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}U^{m+1}_{R(j,k+1)}\right]\right.\\
&\left.+\left[\frac{\cos(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}U^{m+1}_{I(j,k-1)}+\frac{\sin(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k)}\right)^2\right)^2}U^{m+1}_{R(j,k-1)}\right]\right\},
\end{aligned}
$$

e as restantes derivadas de interesse para alimentar as restantes diagonais do bloco:

$$
\begin{aligned}
\frac{\partial F_{I(j,k)}}{\partial U^{m+1}_{I(j+1,k)}}=&\frac{h_t}{h_1^2}\left[-\frac{\cos(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j+1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j+1,k)}\right)^2\right)^2}\right]U^{m+1}_{I(j,k)}+\frac{h_t}{h_1^2}\left[-\frac{\sin(\vartheta)\kappa^2\vartheta^2U^{m+1}_{I(j+1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j+1,k)}\right)^2\right)^2}\right]U^{m+1}_{R(j,k)}\\
&-\frac{h_t}{2h_1^2}\left[\frac{-\cos(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j+1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j+1,k)}\right)^2\right)^2}\left(U^{m+1}_{I(j+1,k)}\right)+\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j+1,k)}}{\kappa\vartheta}\right)^2}+\frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2}-\right.\\
&\left.\frac{\sin(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j+1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j+1,k)}\right)^2\right)^2}\left(U^{m+1}_{R(j+1,k)}\right)\right],
\end{aligned}
$$

$$\frac{\partial F_{I(j,k)}}{\partial U^{m+1}_{I(j-1,k)}} = \frac{h_t}{h_1^2}\left[-\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j-1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j-1,k)}\right)^2\right)^2}\right]U^{m+1}_{I(j,k)} + \frac{h_t}{h_1^2}\left[-\frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j-1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j-1,k)}\right)^2\right)^2}\right]U^{m+1}_{R(j,k)}$$
$$-\frac{h_t}{2h_1^2}\left[\frac{-\cos(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j-1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j-1,k)}\right)^2\right)^2}U^{m+1}_{I(j-1,k)} + \frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j-1,k)}}{\kappa\vartheta}\right)^2} + \frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} - \right.$$
$$\left.\frac{\sin(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j-1,k)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j-1,k)}\right)^2\right)^2}\left(U^{m+1}_{R(j-1,k)}\right)\right],$$

$$\frac{\partial F_{I(j,k)}}{\partial U^{m+1}_{I(j,k+1)}} = \frac{h_t}{h_2^2}\left[-\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k+1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k+1)}\right)^2\right)^2}\right]U^{m+1}_{I(j,k)} + \frac{h_t}{h_2^2}\left[-\frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k+1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k+1)}\right)^2\right)^2}\right]U^{m+1}_{R(j,k)}$$
$$-\frac{h_t}{2h_2^2}\left[\frac{-\cos(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j,k+1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k+1)}\right)^2\right)^2}\left(U^{m+1}_{I(j,k+1)}\right) + \frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k+1)}}{\kappa\vartheta}\right)^2} + \frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} - \right.$$
$$\left.\frac{\sin(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j,k+1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k+1)}\right)^2\right)^2}\left(U^{m+1}_{R(j,k+1)}\right)\right],$$

$$\frac{\partial F_{I(j,k)}}{\partial U^{m+1}_{I(j,k-1)}} = \frac{h_t}{h_2^2}\left[-\frac{\cos(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k-1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k-1)}\right)^2\right)^2}\right]U^{m+1}_{I(j,k)} + \frac{h_t}{h_2^2}\left[-\frac{\sin(\vartheta)\kappa^2\vartheta^2 U^{m+1}_{I(j,k-1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k-1)}\right)^2\right)^2}\right]U^{m+1}_{R(j,k)}$$
$$-\frac{h_t}{2h_2^2}\left[\frac{-\cos(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j,k-1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k-1)}\right)^2\right)^2}\left(U^{m+1}_{I(j,k-1)}\right) + \frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k-1)}}{\kappa\vartheta}\right)^2} + \frac{\cos(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} \right.$$
$$\left.-\frac{\sin(\vartheta)\kappa^2\vartheta^2 2U^{m+1}_{I(j,k-1)}}{\left(\kappa^2\vartheta^2+\left(U^{m+1}_{I(j,k-1)}\right)^2\right)^2}\left(U^{m+1}_{R(j,k-1)}\right)\right],$$

e as derivadas da parte imaginária da função F_I em relação à parte real da imagem U_R

$$\frac{\partial F_{I(j,k)}}{\partial U^{m+1}_{R(j,k)}} = \left\{ \frac{h_t}{2h_1^2} \left[\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j+1,k)}}{\kappa\vartheta}\right)^2} + \frac{2\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} + \frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j-1,k)}}{\kappa\vartheta}\right)^2} \right] + \right.$$
$$\left. + \frac{h_t}{2h_2^2} \left[\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k+1)}}{\kappa\vartheta}\right)^2} + \frac{2\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} + \frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k-1)}}{\kappa\vartheta}\right)^2} \right] \right\},$$

$$\frac{\partial F_{I(j+1,k)}}{\partial U^{m+1}_{R(j+1,k)}} = -\frac{h_t}{2h_1^2} \left[\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j+1,k)}}{\kappa\vartheta}\right)^2} + \frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} \right],$$

$$\frac{\partial F_{I(j-1,k)}}{\partial U^{m+1}_{R(j-1,k)}} = -\frac{h_t}{2h_1^2} \left[\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j-1,k)}}{\kappa\vartheta}\right)^2} + \frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} \right],$$

$$\frac{\partial F_{I(j,k+1)}}{\partial U^{m+1}_{R(j,k+1)}} = -\frac{h_t}{2h_2^2} \left[\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k+1)}}{\kappa\vartheta}\right)^2} + \frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} \right],$$

$$\frac{\partial F_{I(j,k-1)}}{\partial U^{m+1}_{R(j,k-1)}} = -\frac{h_t}{2h_2^2} \left[\frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k-1)}}{\kappa\vartheta}\right)^2} + \frac{\sin(\vartheta)}{1+\left(\frac{U^{m+1}_{I(j,k)}}{\kappa\vartheta}\right)^2} \right].$$

Capítulo 3

Resultados

A fim de executar testes de diminuição de ruído em imagens, aplicando difusão complexa pelos métodos semi-implícito e implícito, utilizou-se a imagem da Lena com dimensões 200×200, que é provavelmente a imagem mais utilizada mundialmente em pesquisa de processamento de imagens.

As imagens plotadas nesse capítulo referem-se a parte real da imagem, uma vez que a parte imaginária contém somente informação sobre os contornos, como pode ser observado na figura 3.1, e como esperado da expressão 1.1.4.

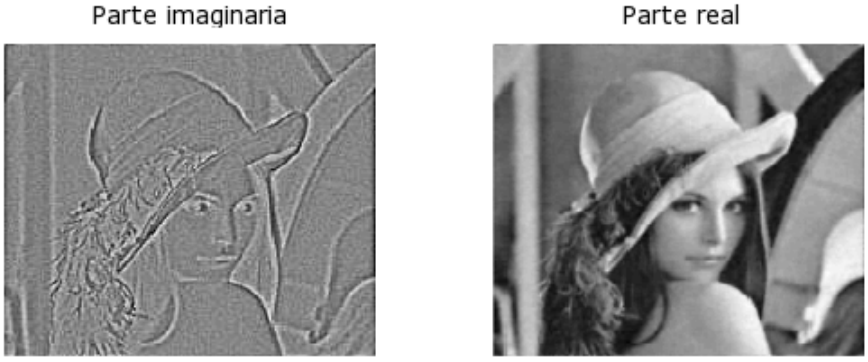

Figura 3.1: Parte imaginária e parte real de uma imagem Lena após filtragem

Por fim, aplicou-se os algoritmos criados neste trabalho às imagens médicas reais, da retina humana, obtidas por tomografia de coerência óptica (OCT). Dessas imagens, foi utilizada uma parte a partir do pixel 150 até o 350 de altura, e do pixel 150 até o 400 de largura, formando uma imagem 200×250. Não se considerou toda a imagem, pois apenas teve-se recursos a computadores pessoais, pelo que a capacidade de memória e velocidade de processamento era

reduzida. Desta forma, pretende-se ilustrar a capacidade dos métodos abordados para retirar ruído em imagens médicas, sem alterar os aspectos de interesse médico da imagem.

Todos os testes foram executados num computador portátil, com processador Intel i3 e 4 gb de memória RAM. Utilizou-se a versão 3.6.1 do Octave para a implementação de todos os algoritmos. Os valores de $\vartheta = \pi/180$ e $\kappa = 10$ foram utilizados na definição do coeficiente de difusão D.

3.1 Semi-implícito

A seguir serão ilustrados os resultados de remoção de ruído com o método semi-implícito.

3.1.1 Dirichlet

Imagem de Teste

Na figura 3.2 pode-se observar a imagem original U, a mesma imagem após adicionar um ruído aleatório (segundo uma distribuição normal), e por fim, a imagem depois de aplicado o filtro por difusão complexa, após o tempo de difusão $T = 0.1$, $T = 0.3$ e $T = 0.5$ utilizando $h_t = 0.1$. O tempo de execução desse algoritmo foi de 7 minutos e 10 segundos para $T = 0.5$ efetuando 5 passos.

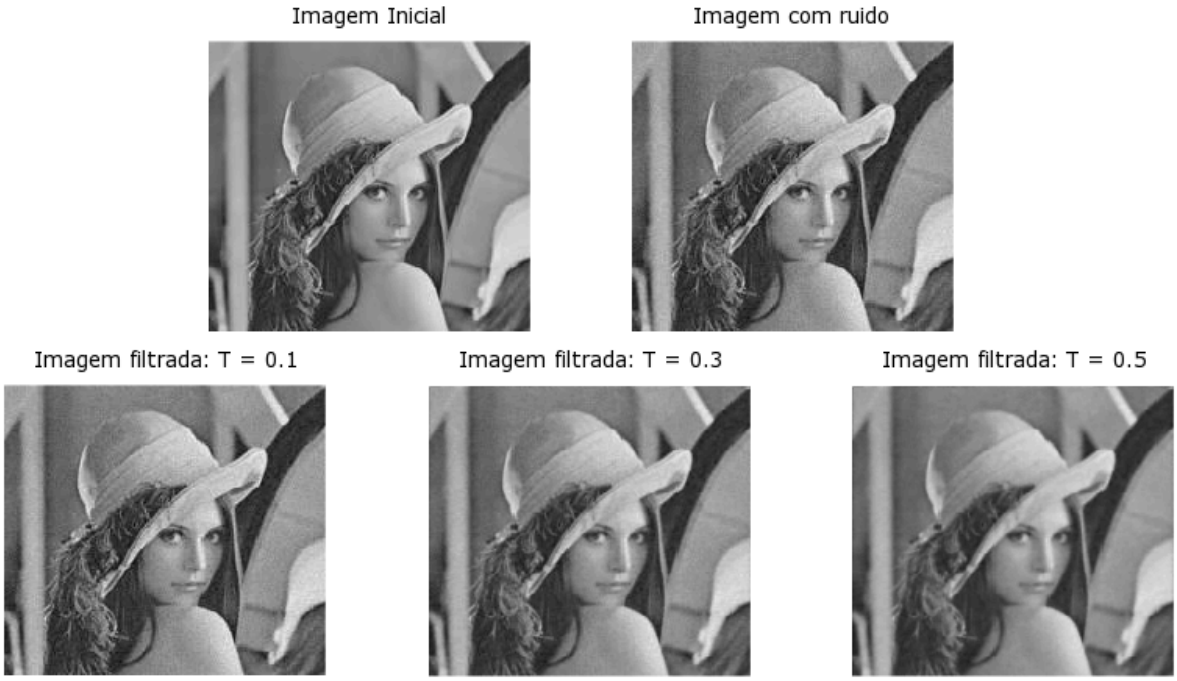

Figura 3.2: Exemplo de imagem tratada - semi-implícito - Dirichlet - $h_t = 0.1$

Já na figura 3.3 observa-se a mesma imagem para um h_t menor, no caso $h_t = 0.05$.
O tempo de execução desse algoritmo foi de 9 minutos e 54 segundos para $T = 0.5$ efetuando 10 passos.

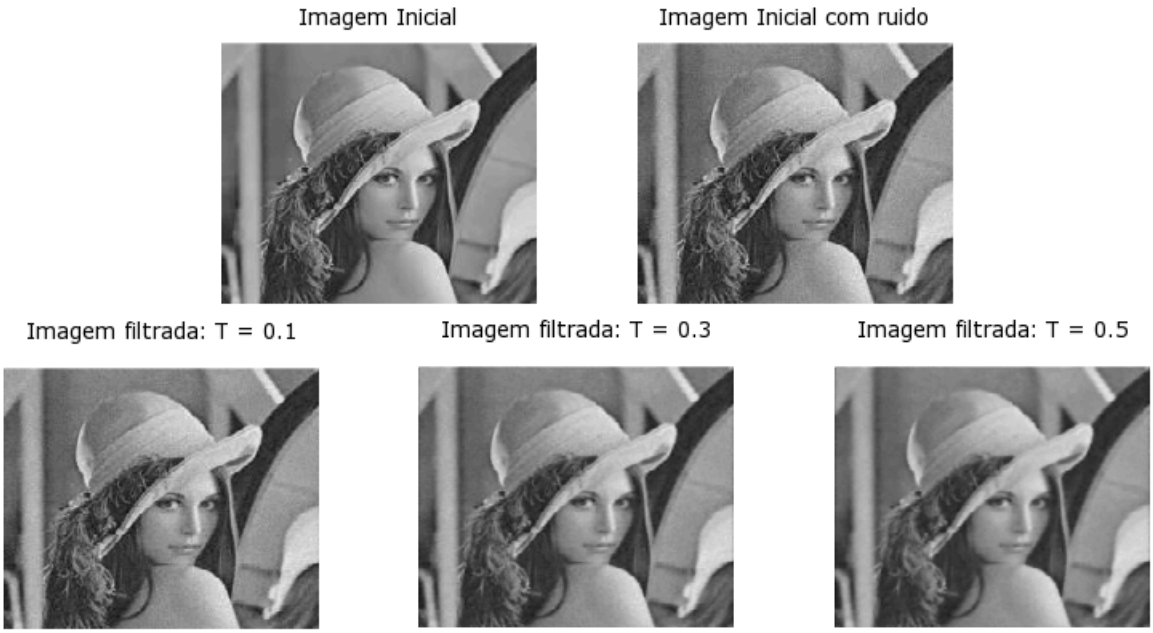

Figura 3.3: Exemplo de imagem tratada - semi-implícito - Dirichlet - $h_t = 0.05$

Por fim, na figura 3.4 tem-se um h_t ainda menor, $h_t = 0.025$, nesse caso pode-se observar menos ruídos na imagem, porém, quanto menor o h_t mais lenta será a execução do algoritmo, pois são necessários mais passos.

Figura 3.4: Exemplo de imagem tratada - semi-implícito - Dirichlet - $h_t = 0.025$

O tempo de execução desse algoritmo foi de 14 minutos e 39 segundos para $T = 0.5$ efetuando 20 passos.

Imagem OCT

A aplicação do algoritmo semi-implícito, com a condição de fronteira de Dirichlet para imagens OCT, pode ser visualizada na figura 3.5, onde a mesma imagem é exibida com diferentes tempos T de difusão. Note que quanto maior o tempo de difusão, mais nítida se torna a imagem, sem se perder a informação relevante das várias camadas da retina.

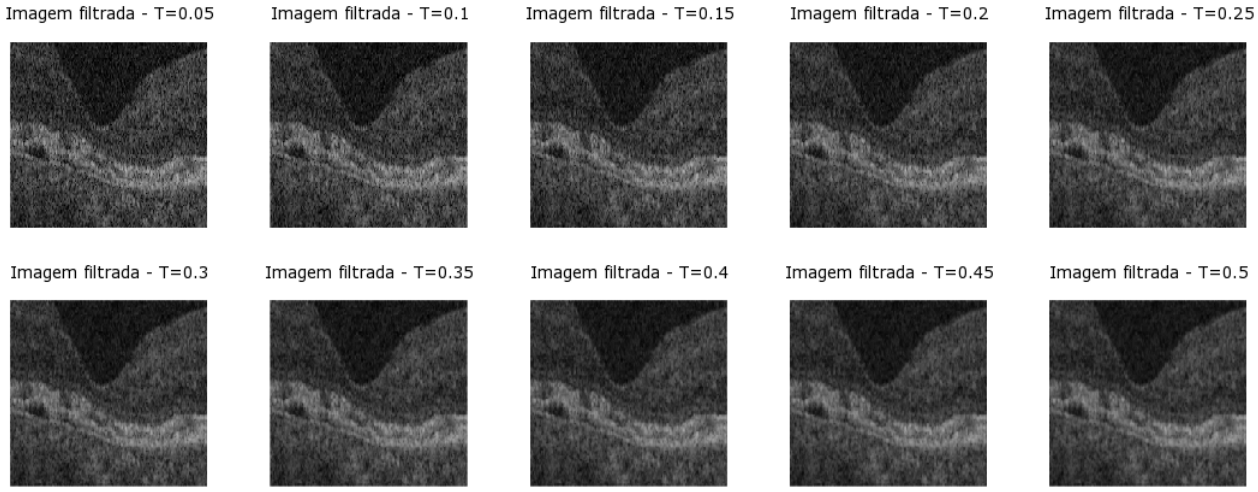

Figura 3.5: Exemplo de imagem OCT tratada - semi-implícito - Dirichlet - $h_t = 0.05$

O tempo de execução desse algoritmo foi de 19 minutos e 16 segundos para $T = 0.5$ e $h_t = 0.05$ efetuando 10 passos.

3.1.2 Neumann

Segue-se agora para o caso da condição de fronteira de Neumann.

Imagem de Teste

Na figura 3.6 pode-se observar a imagem original U, a mesma imagem após adicionar um ruído aleatório (segundo uma distribuição normal), e por fim, a imagem depois de aplicado o filtro por difusão complexa, após o tempo de difusão $T = 0.1$, $T = 0.3$ e $T = 0.5$ utilizando $h_t = 0.05$. O tempo de execução desse algoritmo para $h_t = 0.05$ e $T = 0.5$ foi de 8 minutos e 28 segundos efetuando 10 passos.

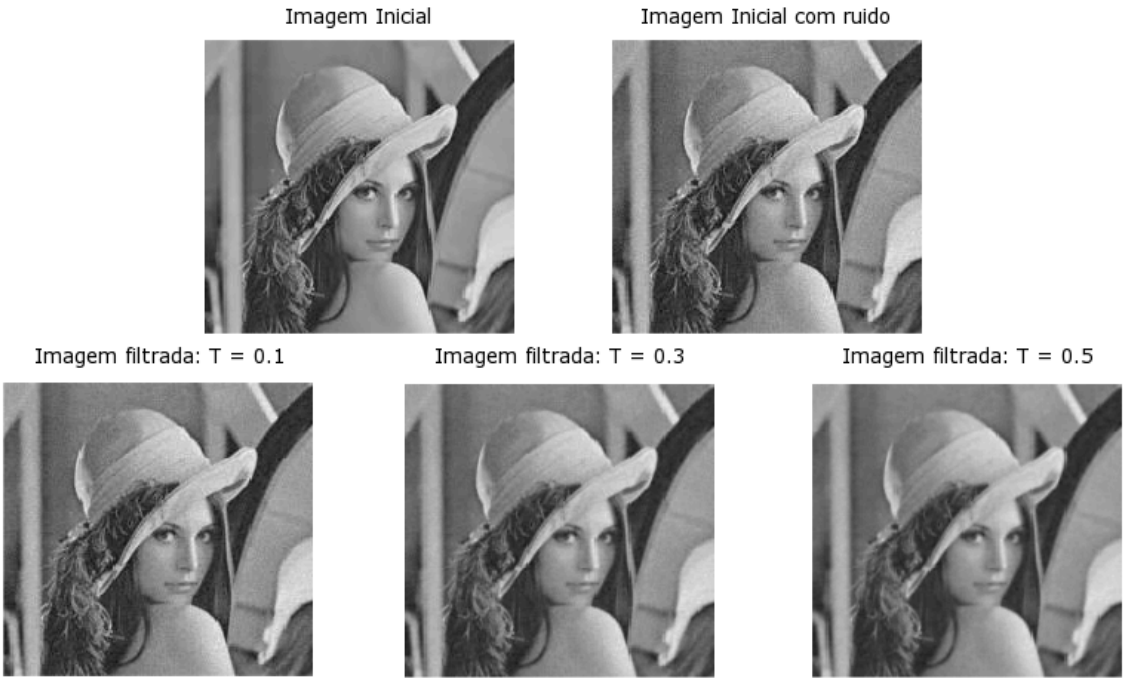

Figura 3.6: Exemplo de imagem tratada - semi-implícito - Neumann - $h_t = 0.05$

Imagem OCT

Assim como para Dirichlet, o algoritmo semi-implícito também foi implementado para a condição de fronteira de Neumann e aplicado em imagens OCT, o que pode ser visualizado na figura 3.7, onde a mesma imagem é exibida com diferentes tempos T de difusão. Note que, quanto maior o tempo de difusão mais nítida se torna a imagem.

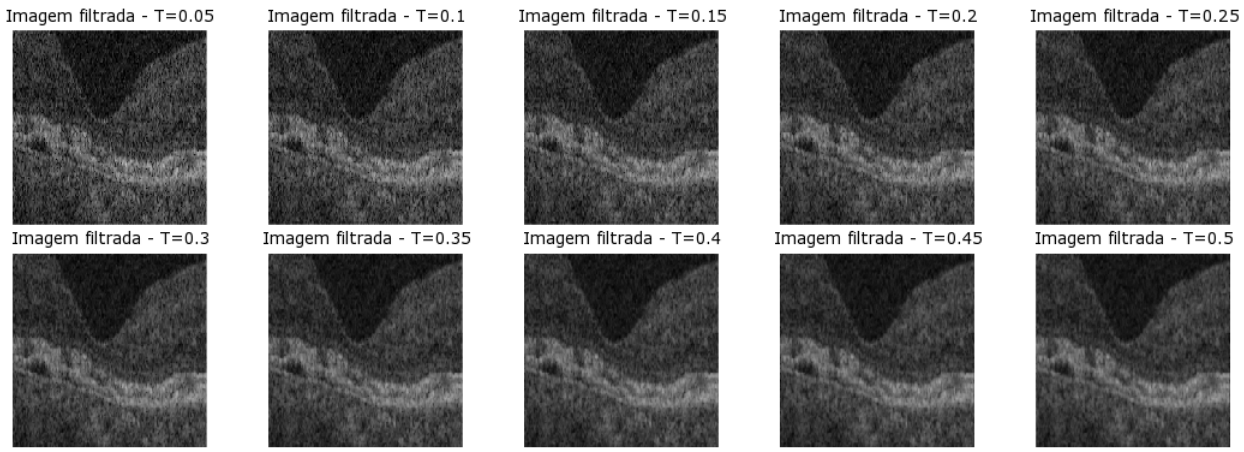

Figura 3.7: Exemplo de imagem OCT tratada - semi-implícito - Neumann - $h_t = 0.05$

O tempo de execução desse algoritmo foi de 1 hora, 39 minutos e 8 segundos para $T = 0.5$ e $h_t = 0.05$ efetuando 10 passos.

3.2 Implícito

No método implícito foi implementado somente a condição de fronteira de Dirichlet, pois não foram detectadas grandes diferenças entre o método semi-implícito de Dirichlet e Neumann.

Lembrando que o critério de paragem para o método implícito foi de T/h_t, por exemplo, para $T = 0.5$ e $h_t = 0.05$ tem-se 10 passos. Já para o método de Newton, definiu-se que a norma L^2 discreta, da diferença entre iteradas consecutivas, seja menor que uma tolerância pré-definida (no caso foi de 10^{-7}). Assim, teve-se a aproximação no instante de tempo t_{m+1} e pôde-se repetir o processo até o tempo de difusão T predefinido.

Também foi criado um número máximo de iterações predefinido, no caso foi utilizado 20, ou seja, se a estrutura de repetição do método de Newton não parar pelo primeiro critério, irá parar por esse segundo, e o sistema avisará que a repetição atingiu o número máximo de repetições.

Imagem Teste

Utilizou-se também a figura da Lena para os testes do método implícito.

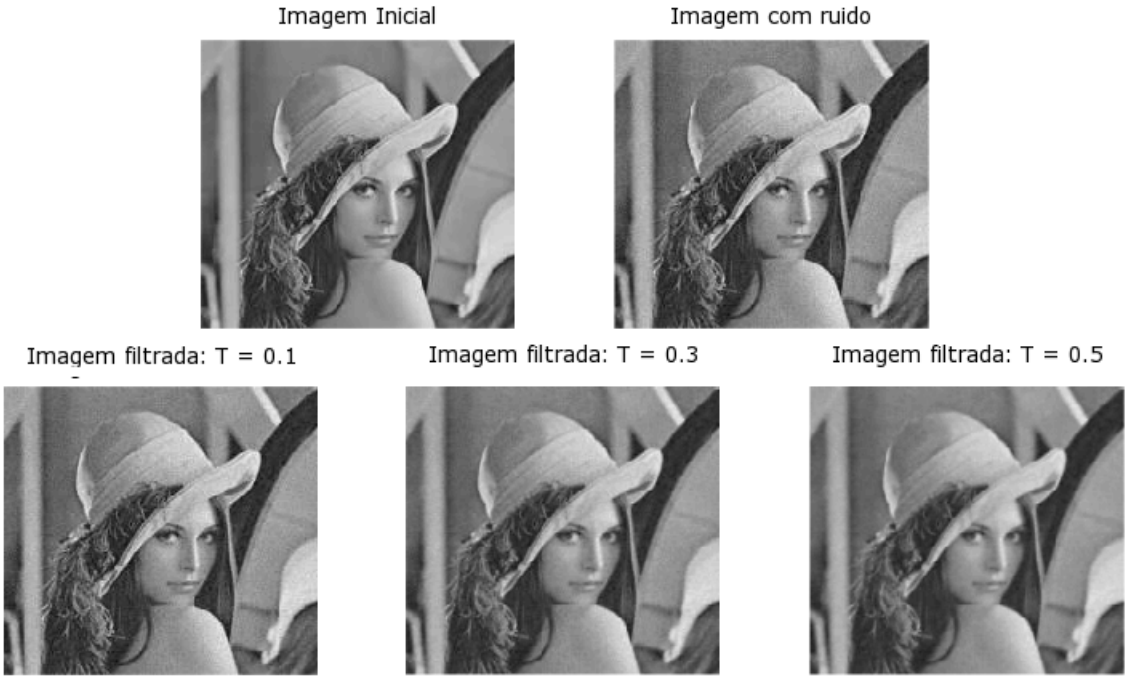

Figura 3.8: Exemplos de imagens tratadas - método implícito - Dirichlet - $h_t = 0.1$

Ilustra-se na figura 3.8 uma imagem original, a mesma imagem após adicionar um ruído aleatório, e uma imagem depois de aplicado o filtro por difusão complexa.

Foi considerada a condição de fronteira de Dirichlet, após o tempo de difusão $T = 0.1$, $T = 0.3$ e por fim $T = 0.5$ utilizando $h_t = 0.1$. Para $T = 0.1$ foi executado 1 passo com 7 iterações do método de Newton.

Na figura 3.9 pode-se observar a imagem original U, a mesma imagem após adicionar um ruído aleatório (segundo uma distribuição normal), e uma imagem depois de aplicado o filtro por difusão complexa, aplicando a condição de fronteira de Dirichlet, após o tempo de difusão $T = 0.1$, $T = 0.3$ e por fim $T = 0.5$ utilizando $h_t = 0.05$. Para $T = 0.5$ foram executados 3 passos com 6 iterações do método de Newton, para o passo 2 e para o passo 3.

Figura 3.9: Exemplos de imagens tratadas - método implícito - Dirichlet - $h_t = 0.05$

Por fim, na figura 3.10 pode-se observar a imagem original U, a mesma imagem após adicionar um ruído aleatório, e uma imagem depois de aplicado o filtro por difusão complexa.

Figura 3.10: Exemplos de imagens tratadas - método implícito - Dirichlet - $h_t = 0.025$

Nessa imagem foi aplicado o método implícito com a condição de fronteira de Dirichlet, após o tempo de difusão $T = 0.1$, $T = 0.3$ e por fim $T = 0.5$ utilizando $h_t = 0.025$.

Veja na relação abaixo a quantidade de passos e as respectivas iterações do método de Newton, o passo 2 refere-se ao $T = 0.1$, o passo 6 refere-se ao $T = 0.3$ e por fim o último passo refere-se ao $T = 0.5$.

Passo número: 1 - Iterações: 6

Passo número: 2 - Iterações: 5

Passo número: 3 - Iterações: 5

Passo número: 4 - Iterações: 5

Passo número: 5 - Iterações: 5

Passo número: 6 - Iterações: 5

Passo número: 7 - Iterações: 5

Passo número: 8 - Iterações: 5

Passo número: 9 - Iterações: 5

Passo número: 10 - Iterações: 5

O tempo total de execução para os 10 passos foi de 10 horas, 45 minutos e 53 segundos.

Para uma melhor ilustração do resultado do método, são apresentados os perfis das imagens para uma melhor comparação. Observe na figura 3.11 a exibição da imagem Lena sem ruído,

após adicionado ruído e após filtragem. O filtro aplicado foi o do método implícito com a condição de fronteira de Dirichlet, $h_t = 0.05$ e $T = 0.3$.

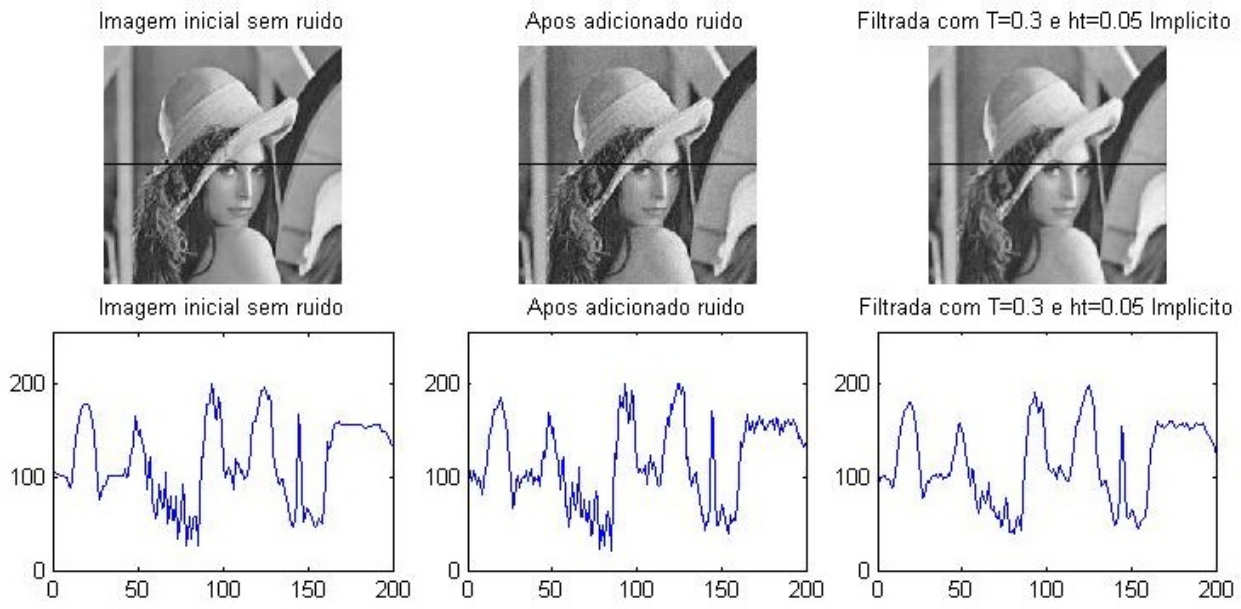

Figura 3.11: Perfil de uma linha da Lena em diferentes estágios

Do mesmo modo pode-se observar na figura 3.12 a diferença entre a imagem sem ruído, após adicionado ruído e após filtragem.

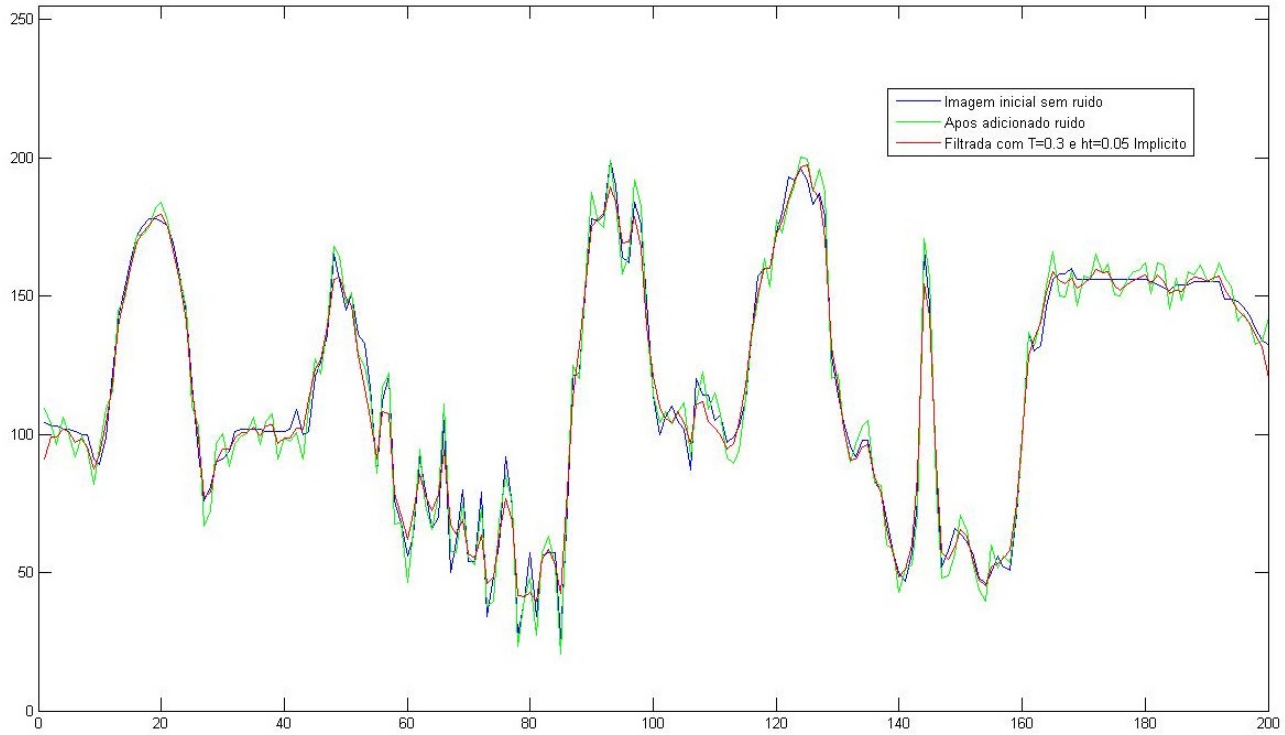

Figura 3.12: Perfil de uma linha da imagem Lena

É visivel que o perfil da imagem filtrada é mais suave do que o da imagem com ruído sem alterar as transições rápidas de intensidade da imagem inicial, como pretendido.

Imagem OCT

Para a imagem OCT foi utilizada uma subárea de uma imagem médica da retina humana. Nessa imagem, foi aplicado o algoritmo implícito e após o tempo de difusão $T = 0.5$ com $h_t = 0.05$ obteve-se o que pode ser observado na figura 3.13

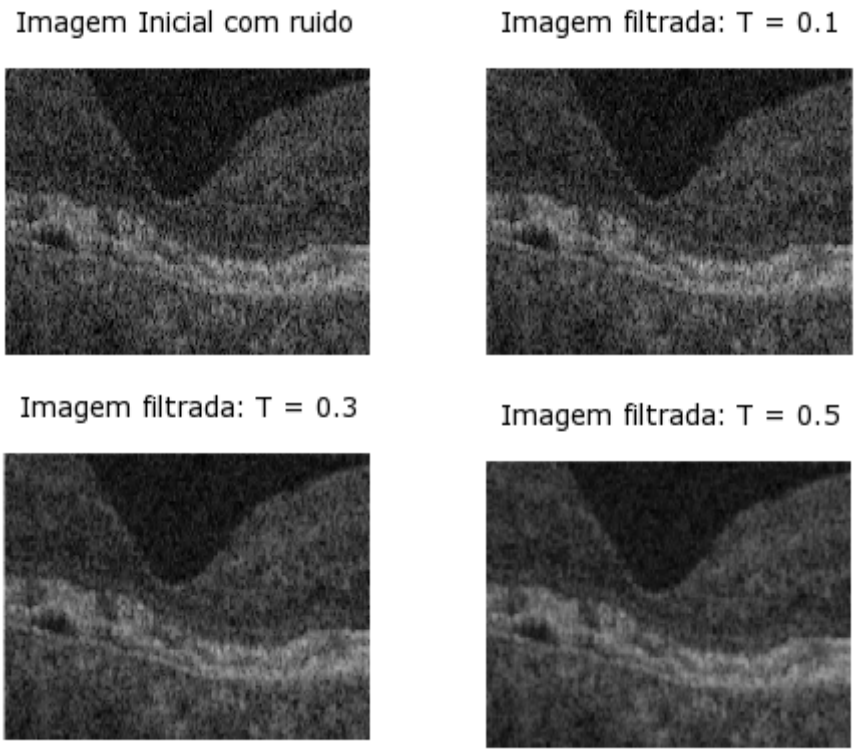

Figura 3.13: Exemplo de imagem OCT tratada - método implícito - Dirichlet - $h_t = 0.05$

Na relação abaixo observe a quantidade de passos e as respectivas iterações do método de Newton, o passo 2 refere-se ao $T = 0.1$, o passo 6 refere-se ao $T = 0.3$ e por fim, o último passo refere-se ao $T = 0.5$.

Passo número: 1 - Iterações: 2

Passo número: 2 - Iterações: 2

Passo número: 3 - Iterações: 2

Passo número: 4 - Iterações: 2

Passo número: 5 - Iterações: 2

Passo número: 6 - Iterações: 2

Passo número: 7 - Iterações: 2

Passo número: 8 - Iterações: 2

Passo número: 9 - Iterações: 2

Passo número: 10 - Iterações: 2

O tempo total de execução para esses passos foi de 9 horas, 40 segundos e 32 segundos.

3.3 Comparações entre Métodos

Pode se observar na figura 3.14, uma comparação entre os resultados da filtragem da imagem Lena a partir dos 3 métodos (semi-implícito Dirichlet, semi-implícito Neumann e implícito Dirichlet) para um mesmo h_t e T. Considerou-se $h_t = 0.05$ e $T = 0.5$ para essa comparação.

Figura 3.14: Exemplo de imagem tratada por diferentes métodos $T = 0.5$ e $h_t = 0.05$

Na figura 3.15, pode-se observar uma comparação entre os resultados da filtragem da imagem Lena para um mesmo método (semi-implícito Dirichlet), para um mesmo T com diferentes valores para h_t. Considerou-se $T = 0.5$ para essa comparação.

Figura 3.15: Exemplo de imagem tratada pelo método semi-implícito Dirichlet - $T = 0.5$

Idealmente a diferença entre a imagem com ruído e a imagem filtrada deveria ser apenas ruído, da mesma forma que a diferença entre a imagem original sem ruído e a imagem filtrada deveria idealmente ser nula. Essas diferenças permitem ilustrar a performance do método. No caso das imagens ilustradas, além da eliminação de ruído, vê-se que existem também algumas diferenças nas variações de intensidade da imagem.

3.3.1 Diferenças Absolutas do Método Semi-implícito - Dirichlet

A seguir serão exibidas imagens de diferenças absolutas do método semi-implícito com a condição de fronteira de Dirichlet, em ambas as imagens foi utilizado $h_t = 0.05$ e $T = 0.3$. Na figura 3.16 é possível observar a diferença entre a imagem Lena depois de adicionado o ruído aleatório e a mesma imagem após filtrada.

Figura 3.16: Diferença absoluta: imagem com ruído e após tratada semi-implícito Dirichlet

Já na figura 3.17 o que pode ser observado é a diferença entre a imagem Lena sem ruído e a mesma após ter adicionado o ruído e filtrada.

Figura 3.17: Diferença absoluta: imagem sem ruído e após tratada semi-implícito Dirichlet

Por fim, na figura 3.18 o que pode ser observado é a diferença entre uma imagem OCT inicial e a mesma após ter sido filtrada.

Figura 3.18: Diferença absoluta: imagem OCT inicial e após tratada semi-implícito Dirichlet

Note que para ambos os casos a diferença é basicamente ruído, que por sua vez é intensificado nos contornos devido à ligeira atenuação dos mesmos pelo processo de difusão. Note-se que nas imagens OCT a diferença é praticamente ruído (os contornos não são muito visíveis na diferença) o que ilustra bem a capacidade de remoção de ruído por esta técnica para este tipo de imagem.

3.3.2 Diferenças Absolutas do Método Semi-implícito - Neumann

Do mesmo modo, para o método semi-implícito com a condição de fronteira de Neumann, pode-se observar nas figuras a seguir a diferença absoluta entre imagens antes e depois de filtradas, em ambas as imagens foi utilizado $h_t = 0.05$ e $T = 0.3$. Na figura 3.19 pode-se observar a diferença entre a imagem da Lena com ruído aleatório e a mesma após filtrada.

Figura 3.19: Diferença absoluta: imagem com ruído e após tratada semi-implícito Neumann

Na figura 3.20 a mesma diferença absoluta é mostrada, porém agora com uma imagem da Lena sem ruído versus a mesma imagem após ter sido adicionado um ruído aleatório e filtrada.

Figura 3.20: Diferença absoluta: imagem sem ruído e após tratada semi-implícito Neumann

Para uma imagem médica OCT, pode-se observar na figura 3.21 a diferença absoluta entre uma imagem inicial (com ruído) e a mesma imagem após tratada.

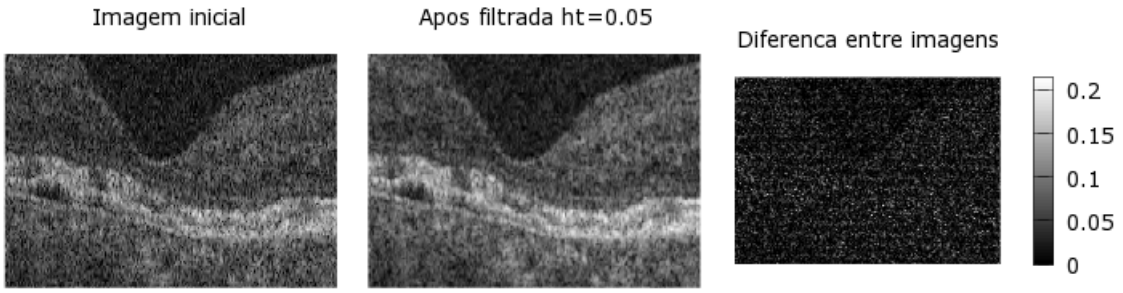

Figura 3.21: Diferença absoluta: imagem OCT inicial e após tratada semi-implícito Neumann

Basicamente a diferença absoluta entre elas é o ruído que é mais intenso nos contornos, note que na imagem 3.21 existe mais diferença de ruído do que na imagem da Lena.

3.3.3 Diferenças Absolutas do Método Implícito

Assim como nos outros métodos, para o método implícito também foi realizada a diferença absoluta entre imagens iniciais e após filtrada. Observe na figura 3.22 a imagem da Lena com ruído aleatório e a mesma imagem após filtrada pelo método implícito com $h_t = 0.05$ e $T = 0.3$, e por fim a diferença absoluta entre elas, que basicamente são os ruídos.

Figura 3.22: Diferença absoluta: imagem com ruído e após tratada - método implícito

No caso da figura 3.23 a imagem inicial utilizada não possui ruído.

Figura 3.23: Diferença absoluta: imagem sem ruído e após tratada - método implícito

No caso de imagem médica 3.24, somente existe a imagem inicial, que por sua vez possui ruído.

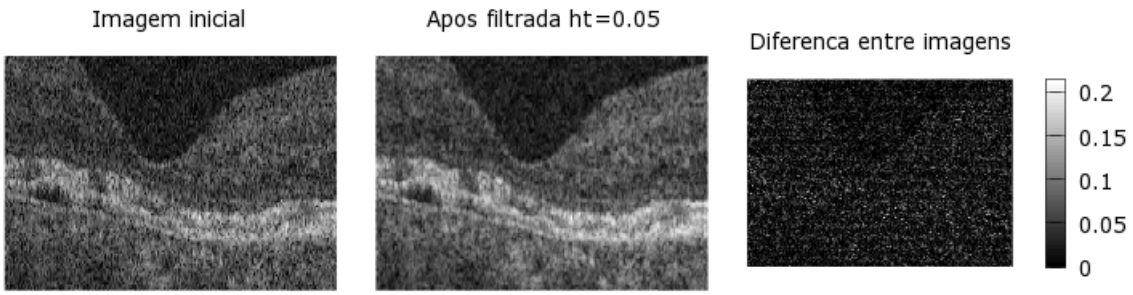

Figura 3.24: Diferença absoluta: imagem OCT inicial e após tratada - método implícito

Em ambos os casos, o que pode ser observado é o mesmo dos outros métodos, a diferença absoluta são ruídos.

Outra observação importante é que no caso das imagens da Lena, fica claro na comparação, que a diferença absoluta entre a imagem sem ruído e a imagem com ruído filtrada, é maior que, a diferença entre a imagem com ruído e a mesma após filtrada.

3.3.4 Métricas Quantitativas

A fim de estudar de forma quantitativa a performance do método, utilizou-se as seguintes métricas:

Erro Quadrático Médio - MSE (*Mean Squared Error*)

$$\text{MSE} = \frac{1}{N_1 N_2} \sum_{j=1}^{N_1} \sum_{k=1}^{N_2} \left(I_{(j,k)} - U_{(j,k)}^{N_t} \right)^2 \tag{3.3.0}$$

Relação Sinal-Ruído de Pico - PSNR (*Peak Signal-to-Noise Ratio*)

$$\text{PSNR} = 10.\log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{3.3.0}$$

Índice de Similaridade Estrutural - MSSIM (*Structural Similarity*)

$$\text{MSSIM} = \frac{1}{N} \sum_{j,k} SSIM\left(\vartheta_I(j,k), \vartheta_U(j,k)\right) \tag{3.3.0}$$

onde

$$\text{SSIM}\left(\vartheta_1, \vartheta_2\right) = \frac{(2\mu_1\mu_2 + C_1)(2\sigma_{12} + C_2)}{(\mu_1^2 + \mu_2^2 + C_1)(\sigma_1^2 + \sigma_2^2 + C_2)} \tag{3.3.0}$$

Desta forma, para $T = 0.3$ obteve-se a tabela 3.1.

Tabela 3.1: Métricas dos métodos com $h_t = 0.05$ e $T = 0.3$

Métricas entre diferentes métodos com ht=0.05 e T=0.3				
Método	Fronteira	MSE	PSNR	MSSIM
Imagem Lena sem ruído e filtrada após adicionado ruído				
Semi-implícito	Dirichlet	26.889	33.869	0.935
	Neumann	24.857	34.210	0.934
Implícito	Dirichlet	23.772	34.404	0.935
Imagem Lena sem ruído e após adicionado ruído				
-	-	33.402	32.927	0.856
Imagem OCT com ruído e filtrada após adicionado ruído				
Semi-implícito	Dirichlet	86.726	28.782	0.829
	Neumann	85.610	28.839	0.829
Implícito	Dirichlet	67.932	29.843	0.867

Observa-se no caso da imagem Lena, cuja se tem a imagem original sem ruído, que o MSE entre as imagens com e sem ruído é maior que entre as imagens sem ruído e filtrada. Isso mostra que o filtro elimina ruído da imagem. Além disso, como o MSSIM é maior no segundo caso do que no primeiro é também claro que a remoção de ruído pelo filtro não altera a estrutura da imagem. Do mesmo modo, o PSNR melhora após a filtragem. Nesse caso, para essa imagem o método implícito apresenta resultados ligeiramente melhores do que o método implícito.

Da mesma forma, para a imagem OCT uma vez que não se tem a imagem sem ruído as métricas são apenas indicativas, no entanto é possível observar a imagem alterada possui um MSE elevado porém sem alterações substanciais na estrutura (MSSIM próximo a 1).

E para $T = 0.5$ obteve-se a tabela 3.2.

Tabela 3.2: Métricas dos métodos com $h_t = 0.05$ e $T = 0.5$

Métricas entre diferentes métodos com ht=0.05 e T=0.5				
Método	Fronteira	MSE	PSNR	MSSIM
Imagem Lena sem ruído e filtrada após adicionado ruído				
Semi-implícito	Dirichlet	44.256	31.704	0.928
	Neumann	41.948	31.937	0.926
Implícito	Dirichlet	40.268	32.115	0.927
Imagem Lena sem ruído e após adicionado ruído				
-	-	33.402	32.927	0.856
Imagem OCT com ruído e filtrada após adicionado ruído				
Semi-implícito	Dirichlet	145.991	26.521	0.691
	Neumann	143.814	26.586	0.691
Implícito	Dirichlet	130.090	27.022	0.722

Em comparação com o exemplo da tabela 3.1 ve-se que algumas métricas são prejudicadas. Isso mostra que a escolha de T é importante para um compromisso entre a eliminação de ruído e o efeito de *blurring*.

Capítulo 4

Conclusão

4.1 Considerações Finais

O tratamento de imagens, foi focado neste trabalho com o desenvolvimento e implementação de um esquema semi-implícito e implícito de diferenças finitas para resolver uma equação diferencial complexa de difusão, para aplicação em filtragem de ruído em imagens. Ao findo desse, pôde-se concluir que a matemática pode auxiliar no tratamento de imagens médicas, como foi o exemplo na aplicação de difusão complexa para filtrar ruídos de imagens da retina humana obtidas por meio de OCT, que possuem bastantes ruídos.

A aplicação, no caso deste trabalho, foi um método de difusão complexa, com as condições de fronteira de Dirichlet e Neumann, o uso do Octave para os experimentos também foi de fundamental importância, pois possui uma linguagem de fácil implementação e adaptada ao cálculo numérico. Porém, por se tratar de sistemas lineares de grande dimensão e de uso de recursos computacionais limitados a um computador pessoal, em alguns casos o processamento foi demasiado lento.

Depois de aprofundados estudos sobre difusão complexa aplicada para filtro de ruído de imagens, passou-se para a etapa de implementação das expressões matemáticas em Octave, durante essa fase do processo, algumas alterações do objetivo inicial foram necessárias, para tornar o algorítmo mais rápido e eficiente, como no caso do método semi-implícito, que não foi preciso usar estruturas de repetição, sendo possível atribuir e calcular os valores das condições de fron-

teiras diretamente na matriz.

Por fim, tem-se 3 três algoritmos, pois foram desenvolvidos os algoritmos para o método semi-implícito com condições de fronteira de Dirichlet e Neumann, sendo que o inicialmente proposto foi somente o método implícito. Os algoritmos para o método semi-implícito são de grande valia, uma vez que o resultado é bastante aproximado do implícito, porém, tem um tempo de execução muito mais rápido.

O método implícito teve uma etapa de desenvolvimento mais complexa, devido à aplicação do método de Newton, foi necessário o cálculo de um Jacobiano trabalhoso, totalizando 20 diferentes derivadas. Os resultados do método implícito, apesar de possuir um cálculo mais lento, a partir das imagens analisadas, aparentemente é muito semelhante aos do método semi-implícito, no caso seria preciso estudos com mais imagens, e uma análise quantitativa da eliminação do ruído para se aferir se algum dos 3 métodos apresentado é significativamente melhor.

Quanto aos resultados da aplicação dos métodos criados, apresentados no capítulo 3, pode-se dizer que foram satisfatórios, tanto para o método semi-implícito quanto para o implícito, uma vez que notoriamente as imagens se mostraram mais nítidas após filtradas. Os cálculos das diferenças absolutas ajudam a clarificar as diferenças entre as imagens iniciais (com e sem ruído) e as imagens após tratadas pelos referidos métodos. De forma geral, as diferenças são essencialmente ruídos com uma maior intensidade nos contornos, devido ao efeito de difusão (ainda que penalizado) que ocorre nas alterações de intensidade da imagem.

4.2 Trabalhos Futuros

Os algoritmos criados por meio deste trabalho não estão otimizados, a propósito, esta não foi uma preocupação durante a fase de desenvolvimento. Portanto em uma etapa seguinte pode-se otimizar os algoritmos apresentados aqui - tanto no âmbito da construção das matrizes dos sistemas, como da sua resolução - em especial a parte de difusão complexa pelo método implícito, que leva bastante tempo para filtrar imagens relativamente pequenas, conforme apresentado nos resultados. Uma sugestão seria utilizar uma abordagem multigrid, ou seja, considerando a imagem com diferentes resoluções, em que esta aumenta à medida da evolução das itera-

ções; outra ideia seria de resolver o sistema linear de forma iterativa, ao invés de resolvê-lo diretamente. Além disso, pode-se sugerir eliminar ao máximo as estruturas de repetição *for*, atribuindo os valores diretamente nas matrizes esparsas consideradas.

Outra sugestão para trabalhos futuros seria a implementação do filtro de ruídos por difusão complexa, pelo método implícito utilizando as condições de fronteira de Neumann, pois devido a limitações de tempo e a diferença não representativa verificada no método semi-implícito, o mesmo não foi desenvolvido neste trabalho. Finalmente, será necessário fazer um estudo quantitativo comparativo quanto a eliminação de ruído e diferenças em relação à imagem inicial sem ruído, para se ter uma ideia da performance de cada método, o que poderá resultar numa submissão para publicação numa revista científica.

Referências Bibliográficas

[Araújo et al., 2012] Araújo, A., Barbeiro, S., and Serranho, P. (2012). Stability of finite difference schemes for complex diffusion processes. *SIAM J. Numer. Anal.*, 50 (3):1284–1296.

[Araújo et al., 2013] Araújo, A., Barbeiro, S., and Serranho, P. (2013). Finite difference schemes for nonlinear complex diffusion processes. *to appear*.

[Bernardes et al., 2010] Bernardes, R., Maduro, C., Serranho, P., Araújo, A., Barbeiro, S., and Cunha-Vaz, J. (2010). Improved adaptive complex diffusion despeckling filter. *Optics Express*, 18 (23):24048–24059.

[Brox et al., 2004] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). *Computer Vision - ECCV 2004. Lecture Notes in Computer Science*, volume 3024, chapter High accuracy optical flow estimation based on a theory for warping, pages 25–36. Springer, Berlin.

[Gilboa et al., 2004] Gilboa, G., Sochen, N., and Zeeni, Y. (2004). Image enhancement and denoising by complex diffusion processes. *IEEE Trans Pattern Anal Mach Intell*, 26 (8):1020–1036.

[Grossauer and Scherzer, 2003] Grossauer, H. and Scherzer, O. (2003). *Scale Space Methods in Computer Vision, Lecture Notes in Computer Science*, volume 2695, chapter Using the Complex Ginzburg-Landau Equation for Digital Inpainting in 2D and 3D, pages 225–236. Springer.

[Perona and Malik, 1990] Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Trans Pattern Anal Mach Intell*, 12 (7):629–639.

[Salinas and Fernández, 2007] Salinas, H. and Fernández, D. (2007). Comparison of PDE-based nonlinear diffusion approaches for image enhancement and denoising in optical coherence tomography. *IEEE Trans. Med. Imaging*, 26 (6):761–771.

[Weickert, 1994] Weickert, J. (1994). Anisotropic diffusion filters for image processing based quality control. *Proc. 7th Eur. Conf. Mathematics in Industry*, 1252:355–362.

[Weickert, 1997] Weickert, J. (1997). A review of nonlinear diffusion filtering. In ter Haar Romeny, B. M., Florack, L., Koenderink, J. J., and Viergever, M. A., editors, *Scale-Space*, volume 1252 of *Lecture Notes in Computer Science*, pages 3–28. Springer.

[Zimmer et al., 2008] Zimmer, H., Bruhn, A., Valgaerts, L., Breuß, M., Weickert, J., Rosenhahn, B., and Seidel, H.-P. (2008). *Vision, Modeling, and Visualization*, chapter PDE-based anisotropic disparity-driven stereo vision, pages 263–272. AKA Heidelberg.

Apêndices

Apêndice A

Algoritmo do método semi-implícito com condição de fronteira de Dirichlet em Octave, para filtragem da imagem Lena.

```
% Semi-implicito DIRICHLET
tic;
theta = pi/180;
kappa = 10;
ht = 0.05; % quanto menor converge melhor
h1 = 1;
h2 = 1;
T = 0.5; % maior T limpa mais muito ruido mas desfoca mais a imagem
nIteracao = round(T/ht);

load('ImL.mat');
ImI = ImL;
load('ImR.mat');
UI = ImR;

U = zeros(size(UI,1),size(UI,2),nIteracao+1);
n1 = size(UI,1);
n2 = size(UI,2);
usize = [size(UI,1),size(UI,2)];
nPontos =  prod(usize);

U(:,:,1) = UI;

% Usado para adicionar um ruido aleatorio a imagem
% U = U + (A*(0.5-rand(size(U))));

% Grande matriz esparsa
MatrizA = sparse(nPontos,nPontos);
VetorB = zeros(n1,n2);

for iteracao = 1 : nIteracao
    D = exp(i*theta)./(1+(imag(U(:,:,iteracao))/kappa/theta).^2);
```

```
33      vD = reshape(D,[nPontos,1]);
34      %k+1
35      DauxKK = D;
36      DauxKK(:,1:end-1) = DauxKK(:,2:end);
37      vDKK = reshape(DauxKK,[nPontos,1]);
38      %k-1
39      DauxK = D;
40      DauxK(:,2:end) = DauxK(:,1:end-1);
41      vDK = reshape(DauxK,[nPontos,1]);
42      %j+1
43      DauxJJ = D;
44      DauxJJ(1:end-1,:) = DauxJJ(2:end,:);
45      vDJJ = reshape(DauxJJ,[nPontos,1]);
46      %j-1
47      DauxJ = D;
48      DauxJ(2:end,:) = DauxJ(1:end-1,:);
49      vDJ = reshape(DauxJ,[nPontos,1]);
50
51      % atribui o vetor da diagonal na grande Matriz
52      for iter = 1 : nPontos
53          MatrizA(iter,iter) = 1 + (ht/2/h1^2)*((vDJJ(iter)) + ...
54          2*vD(iter) + (vDJ(iter))) + (ht/2/h2^2)*((vDKK(iter)) + ...
55          2*vD(iter) + (vDK(iter)));
56      end
57
58      VetorB(:,:) = U(:,:,iteracao);
59
60      %U(k-1)
61      AuxA = (-ht/2/h2^2) * (vD + vDK);
62      A = reshape(AuxA,[n1,n2]);
63      for k = 1 : n2
64         for j = 1 : n1
65             if(k == 1)
66                 VetorB(j,k) = VetorB(j,k) - (A(j,k) * UI(j,k,1));
67                 A(j,k) = 0;
68             end
69        end
70      end
71      AuxA = reshape(A,[nPontos,1]);
```

```
72      for iter = 2 : nPontos
73         MatrizA(iter,iter-1) = AuxA(iter);
74      end
75
76      %U(k+1)
77      AuxA = (-ht/2/h2^2) * (vDKK + vD);
78      A = reshape(AuxA,[n1,n2]);
79      for k = 1 : n2
80         for j = 1 : n1
81             if(k == n2)
82                VetorB(j,k) = VetorB(j,k) - ((A(j,k)) * UI(j,k,1));
83                A(j,k) = 0;
84             end
85         end
86      end
87      AuxA = reshape(A,[nPontos,1]);
88      for iter = 1:nPontos-1
89         MatrizA(iter,iter+1) = AuxA(iter);
90      end
91
92      %U(j+1)
93      AuxA = (-ht/2/h1^2) * (vDJJ + vD);
94      A = reshape(AuxA,[n1,n2]);
95      for k = 1 : n2
96         for j = 1 : n1
97             if(j == n1)
98                VetorB(j,k) = VetorB(j,k) - (A(j,k) * UI(j,k,1));
99                A(j,k) = 0;
100            end
101        end
102     end
103     AuxA = reshape(A,[nPontos,1]);
104     for iter = 1 : nPontos- n1
105        MatrizA(iter,iter+n1) = AuxA(iter);
106     end
107
108     %U(j-1)
109     AuxA = (-ht/2/h2^2) * (vD + vDJ);
110     A = reshape(AuxA,[n1,n2]);
```

```
111     for k = 1 : n2
112        for j = 1 : n1
113            if(j == 1)
114               VetorB(j,k) = VetorB(j,k) - (A(j,k) * UI(j,k,1));
115               A(j,k) = 0;
116            end
117        end
118     end
119     AuxA = reshape(A,[nPontos,1]);
120     for iter = n1+1 : nPontos
121        MatrizA(iter,iter-n1) = AuxA(iter);
122     end
123
124 % VetorB
125 AuxB = reshape(VetorB,[nPontos,1]);
126
127 % Resolve o sistema linear
128 Unovo = MatrizA\AuxB;
129
130 % Reshape do Unovo para recriar uma matriz n1 x n2
131 U(:,:,iteracao+1) = reshape(Unovo,[n1,n2]);
132
133 fprintf('\nPasso: %d', round(iteracao))
134
135 end % for da iteracao
136
137 t = toc;
138 fprintf('\nTempo de execucao: %s \n',datestr(datenum(0,0,0,0,0,t),'HH:MM:SS'))
```

Apêndice B

Algoritmo do método semi-implícito com condição de fronteira de Neumann em Octave, também para filtragem da imagem Lena.

```
% Semi-implicito NEUMANN
tic;
theta = pi/180;
kappa = 10;
ht = 0.05;
h1 = 1;
h2 = 1;
A = 20;
T = 0.5;
nIteracao = round(T/ht);

load('ImL.mat');
ImI = ImL;
load('ImR.mat');
UI = ImR;
U = zeros(size(UI,1),size(UI,2),nPontosacao+1);

n1 = size(UI,1);
n2 = size(UI,2);
usize = [size(UI,1),size(UI,2)];
nPontos= prod(usize);

U(:,:,1) = UI;

% Usado para adicionar um ruido aleatorio a imagem
% U = U + (A*(0.5-rand(size(U))));

% Grande matriz esparsa
MatrizA = sparse(nPontos,nPontos);
VetorB = zeros(n1,n2);

for iteracao = 1 : nIteracao
    D = exp(i*theta)./(1+(imag(U(:,:,iteracao))/(kappa*theta)).^2);
    vD = reshape(D,[nPontos,1]);
```

```
35      %k+1
36      DauxKK = D;
37      DauxKK(:,1:end-1) = DauxKK(:,2:end);
38      vDKK = reshape(DauxKK,[nPontos,1]);
39      %k-1
40      DauxK = D;
41      DauxK(:,2:end) = DauxK(:,1:end-1);
42      vDK = reshape(DauxK,[nPontos,1]);
43      %j+1
44      DauxJJ = D;
45      DauxJJ(1:end-1,:) = DauxJJ(2:end,:);
46      vDJJ = reshape(DauxJJ,[nPontos,1]);
47      %j-1
48      DauxJ = D;
49      DauxJ(2:end,:) = DauxJ(1:end-1,:);
50      vDJ = reshape(DauxJ,[nPontos,1]);
51
52      % Atribui o vetor da diagonal na grande Matriz
53      for iter = 1:nPontos
54          MatrizA(iter,iter) = 1 + (ht/(2*h1^2))*((vDJJ(iter)) + ...
55          2*vD(iter) + (vDJ(iter))) + (ht/(2*h2^2))*((vDKK(iter)) + ...
56          2*vD(iter) + (vDK(iter)));
57      end
58      VetorB(:,:) = U(:,:,iteracao);
59
60      %u-k
61      AuxA = (-ht/(2*h2^2)) * (vD + vDK);
62      A = reshape(AuxA,[n1,n2]);
63      for j = 1 : n1
64         for k = 1 : n2
65             if(k == 1)
66                 A(j,k+1) = A(j,k+1) + A(j,k);
67                 A(j,k) = 0;
68             end
69        end
70      end
71      AuxA = reshape(A,[nPontos,1]);
72      for iter = n1+1 : nPontos
73         MatrizA(iter-n1,iter) = AuxA(iter);
```

```
74      end
75      %u+k
76      AuxA = (-ht/(2*h2^2)) * (vDKK + vD);
77      A = reshape(AuxA,[n1,n2]);
78      for j = 1 : n1
79         for k = 1 : n2
80             if(k == n2)
81                A(j,k-1) = A(j,k-1) + A(j,k);
82                A(j,k) = 0;
83             end
84        end
85      end
86      AuxA = reshape(A,[nPontos,1]);
87      for iter = 1 : nPontos- n1
88         MatrizA(iter+n1,iter) = AuxA(iter);
89      end
90      %u+j
91      AuxA = (-ht/(2*h1^2)) * (vDJJ + vD);
92      A = reshape(AuxA,[n1,n2]);
93      for j = 1 : n1
94         for k = 1 : n2
95             if(j == n1)
96                A(j-1,k) = A(j-1,k)+ A(j,k) ;
97                A(j,k) = 0;
98             end
99         end
100     end
101     AuxA = reshape(A,[nPontos,1]);
102     for iter = 1 : nPontos-1
103        MatrizA(iter+1,iter) = AuxA(iter);
104     end
105     %u-j
106     AuxA = (-ht/(2*h1^2)) * (vD + vDJ);
107     A = reshape(AuxA,[n1,n2]);
108     for j = 1 : n1
109        for k = 1 : n2
110            if(j == 1)
111               A(j+1,k) = A(j+1,k) + A(j,k);
112               A(j,k) = 0;
```

```
113             end
114         end
115     end
116     AuxA = reshape(A,[nPontos,1]);
117     for iter = 2 : nPontos
118         MatrizA(iter-1,iter) = AuxA(iter);
119     end
120
121 % VetorB
122 AuxB = reshape(VetorB,[nPontos,1]);
123
124 % Resolve o sistema linear
125 Unovo = MatrizA\AuxB;
126
127 % Reshape do Unovo para recriar uma matriz n1 x n2
128 U(:,:,iteracao+1) = reshape(Unovo,[n1,n2]);
129
130 fprintf('\nPasso: %d', round(iteracao))
131
132 end % for da iteracao
133 t = toc;
134 fprintf('\nTempo de execucao: %s \n',datestr(datenum(0,0,0,0,0,t),'HH:MM:SS'))
```

Apêndice C

Algoritmo do método implícito com condição de fronteira de Dirichlet em Octave, para imagem médica OCT.

```
% Implicito DIRICHLET
tic;
theta = pi/180;
kappa = 10;
ht = 0.05;
h1 = 1;
h2 = 1;
T = 0.5;
maxIter = 20; % numero maximo de iteracoes
nPassos = round(T/ht);

% Pre calcula alguns valores uzados nas derivadas
st = sin(theta);
ct = cos(theta);
kt2 = (kappa^2 * theta^2);
kt = kappa * theta;

load('ImI5.mat'); % carrega imagem medica
UR = ImI(150:350,150:400); % seleciona parte da imagem OCT
ImI = UR; % atribui novamente a imagem inical ja cortada
UI = imag(UR);

U = zeros(size(UI,1),size(UI,2),nPassos+1);% declara U
U(:,:,1) = UR; % atribui UR para U, x0

n1 = size(UR,1);
n2 = size(UR,2);
usize = [size(UI,1),size(UI,2)];
nPontos=  prod(usize); % numero de iteracoes

URN = real(U(:,:,1));
UIN = imag(U(:,:,1));
UR = URN;
UI = UIN;
```

```
35
36 for passo = 1 : nPassos % inicio do metodo implicito
37     L2dif = 10^(-7); % inicializacao da variavel
38     iteracao = 0;
39
40     while (L2dif > (10^(-8))) && (iteracao < maxIter);
41     iteracao = iteracao+1;
42
43     clear BL1 BL2 BL3 BL4 MatrizFl VectorFU;
44
45     % Declaracao blocos grande matriz
46     BL1 = sparse(nPontos,nPontos);
47     BL2 = sparse(nPontos,nPontos);
48     BL3 = sparse(nPontos,nPontos);
49     BL4 = sparse(nPontos,nPontos);
50     MatrizFl = sparse(2*nPontos,2*nPontos);
51
52     %k+1
53     UauxKK = UR+i.*UI;
54     UauxKK(:,1:end-1) = UauxKK(:,2:end);
55     UauxKK(:,end) = ImI(:,end);
56     uiKK = imag(UauxKK);
57     urKK = real(UauxKK);
58     %k-1
59     UauxK = UR+i.*UI;
60     UauxK(:,2:end) = UauxK(:,1:end-1);
61     UauxK(:,1) = ImI(:,1);
62     uiK = imag(UauxK);
63     urK = real(UauxK);
64     %j+1
65     UauxJJ = UR+i.*UI;
66     UauxJJ(1:end-1,:) = UauxJJ(2:end,:);
67     UauxJJ(:,end) = ImI(:,end);
68     uiJJ = imag(UauxJJ);
69     urJJ = real(UauxJJ);
70     %j-1
71     UauxJ = UR+i.*UI;
72     UauxJ(2:end,:) = UauxJ(1:end-1,:);
73     UauxJ(:,1) = ImI(:,1);
```

```
74      uiJ = imag(UauxJ);
75      urJ = real(UauxJ);
76
77      % IMPLICITO, funcao F, parte real e imaginaria
78      FR = (1+ (ht./(2.*h1.^2)) .* ((ct ./ (1+ (uiJJ).^2 ./ (kt2)))+ ...
79      (2 .* ct ./ (1+ (UI).^2 ./ kt2)) + (ct ./ (1+ (uiJ).^2 ./ kt2)))+ ...
80      ((ht./(2.*h2.^2)) .* ((ct ./ (1+ (uiKK).^2 ./ kt2)) + (2 .* ct ./ ...
81      (1+ (UI).^2 ./ kt2)) + (ct ./ (1+ (uiK).^2 ./ kt2))))).*UR ...
82      - (((ht./(2.*h1.^2)) .* ((st ./ (1+ (uiJJ).^2 ./ kt2)) + (2 .* st ...
83      ./ (1+ (UI).^2 ./ kt2)) + (st ./ (1+ (uiJ).^2 ./ kt2))) + ...
84      ((ht./(2.*h2.^2)) .* ((st ./ (1+ (uiKK).^2 ./ kt2)) + (2 .* st ./ ...
85      (1+ (UI).^2 ./ kt2)) + (st ./ (1+ (uiK).^2 ./ kt2))))).*UI ...
86      - (ht./(2.*h1.^2)) .* (((ct ./ (1+ (uiJJ).^2 ./ kt2)) + ...
87      (ct ./ (1+ (UI).^2 ./ kt2))).*urJJ  - ((st ./ (1+ (uiJJ).^2 ./ ...
88      kt2)) + (st ./ (1+ (UI).^2 ./ kt2))).*uiJJ + ((ct ./ (1+ (UI).^2 ...
89      ./ kt2)) + (ct ./ (1+ (uiJ).^2 ./ kt2))).*urJ - ...
90      ((st ./ (1+ (UI).^2 ./ kt2)) + (st ./ (1+ (uiJ).^2 ./ kt2))).*uiJ) ...
91      - (ht./(2.*h2.^2)) .* ((ct ./ (1+ (uiKK.^2 ./ kt2)) + ...
92      (ct ./ (1+ (UI).^2 ./ kt2))).*urKK - ((st ./ (1+ (uiKK).^2 ./ ...
93      kt2)) + (st ./ (1+ (UI).^2 ./ kt2))).*uiKK  + ((ct ./ (1+ (UI).^2 ...
94      ./ kt2)) + (ct ./ (1+ (uiK).^2 ./ kt2))).*urK - ((st ./ ...
95      (1+ (UI).^2 ./ kt2)) + (st ./ (1+ (uiK).^2 ./ kt2))).*uiK)-URN;
96
97      FI = (1+ (ht./(2.*h1.^2)) .* ((ct ./ (1+ (uiJJ).^2 ./ kt2)) + ...
98      (2 .* ct ./ (1+ (UI).^2 ./ kt2)) + (ct ./ (1+ (uiJ).^2 ./ kt2))) ...
99      + ((ht./(2.*h2.^2)) .* ((ct ./ (1+ (uiKK).^2 ./ kt2)) + (2 .* ct ./ ...
100     (1+ (UI).^2 ./ kt2)) + (ct ./ (1+ (uiK).^2 ./ kt2))))).*UI ...
101     + (((ht./(2.*h1.^2)) .* ((st ./ (1+ (uiJJ).^2 ./ kt2)) + (2 .* st ./ ...
102     (1+ (UI).^2 ./ kt2)) + (st ./ (1+ (uiJ).^2 ./ kt2))) + ...
103     ((ht./(2.*h2.^2)) .* ((st ./ (1+ (uiKK).^2 ./ kt2)) + ...
104     (2 .* st ./ (1+ (UI).^2 ./ kt2)) + (st ./ (1+ (uiK).^2 ./ kt2))))).*UR ...
105     - (ht./(2.*h1.^2)) .* (((ct ./ (1+ (uiJJ).^2 ./ kt2)) + (ct ./ ...
106     (1+ (UI).^2 ./ kt2))).*uiJJ + ((st ./ (1+ (uiJJ).^2 ./ kt2)) + ...
107     (st ./ (1+ (UI).^2 ./ kt2))).*urJJ  + ((ct ./ (1+ (UI).^2 ./ kt2)) + ...
108     (ct ./ (1+ (uiJ).^2 ./ kt2))).*uiJ  + ((st ./ (1+ (UI).^2 ./ kt2)) + ...
109     (st ./ (1+ (uiJ).^2 ./ kt2))).*urJ) ...
110     - (ht./(2.*h2.^2)) .* ((ct ./ (1+ (uiKK.^2 ./ kt2)) + ...
111     (ct ./ (1+ (UI).^2 ./ kt2))).*uiKK  + ((st ./ (1+ (uiKK).^2 ./ ...
112     kt2)) + (st ./ (1+ (UI).^2 ./ kt2))).*urKK  + ((ct ./ (1+ (UI).^2 ./ ...
```

```
kt2)) + (ct ./ (1+ (uiK).^2 ./ kt2))).*uiK  + ((st ./ (1+ (UI).^2 ./ ...
kt2)) + (st ./ (1+ (uiK).^2 ./ kt2))).*urK)-UIN;

% DERIVADAS
FrUi =  ((ht./(h1.^2)) .* ( -(2.*ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2) + ...
(ht./(h2.^2)) .* ( -(2.*ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2)) .* UR ...
- ((ht./(h1.^2)) .* ( -(2.*st.*kt2.*UI) ./ (kt2 + (UI).^2).^2) + ...
(ht./(h2.^2)) .* ( -(2.*st.*kt2.*UI) ./ (kt2 + (UI).^2).^2)) .* UI ...
- ((ht./(2.*h1.^2)) .* ((st ./ (1+ (uiJJ).^2 ./ kt2)) + (2 .* st ./ ...
(1+ (UI).^2 ./ kt2)) + (st ./ (1+ (uiJ).^2 ./ kt2))) + ...
((ht./(2.*h2.^2)) .* ((st ./ (1+ (uiKK).^2 ./ kt2)) + ...
(2 .* st ./ (1+ (UI).^2 ./ kt2)) + (st ./ (1+ (uiK).^2 ./ kt2))))) ...
- (ht./(h1.^2)) .* (( -(ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2) .* urJJ + ...
((st.*kt2.*UI) ./ (kt2 + (UI).^2).^2) .* uiJJ - ((ct.*kt2.*UI) ./ ...
(kt2 + (UI).^2).^2) .* urJ + ((st.*kt2.*UI) ./ (kt2 + (UI).^2).^2) .* ...
uiJ) - (ht./(h2.^2)) .* (( -(ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2) .* ...
urKK + ((st.*kt2.*UI) ./ (kt2 + (UI).^2).^2) .* uiKK - ((ct.*kt2.*UI) ...
./ (kt2 + (UI).^2).^2) .* urK + ((st.*kt2.*UI) ./ (kt2 + (UI).^2).^2) ...
.* uiK );

FrUiJJ = (- ht ./ (h1.^2)) .* ((ct.*kt2.*uiJJ) ./ ...
(kt2 + (uiJJ).^2).^2).*UR   + (ht ./ h1.^2) .* ((st.*kt2.*uiJJ) ./ ...
(kt2 + (uiJJ).^2).^2).*UI - (ht ./ h1.^2) .* ((-(ct.*kt2.*uiJJ) ./ ...
(kt2 + (uiJJ).^2).^2).*urJJ + ((st.*kt2.*uiJJ) ./ (kt2 + (uiJJ).^2) ...
.^2 ).*uiJJ - (1./2) .*     ((st./ (1 + (uiJJ ./ kt).^2)) + ...
( st ./ (1 + (UI./ kt).^2))));

FrUiJ = (- ht ./ (h1.^2)) .* ((ct.*kt2.*uiJ) ./ (kt2 + (uiJ).^2).^2) ...
.*UR + (ht ./ (h1.^2)) .* ((st.*kt2.*uiJ) ./ (kt2 + (uiJ).^2).^2).*UI ...
- (ht ./ (h1.^2)) .* ((-(ct.*kt2.*uiJ) ./ (kt2 + (uiJ).^2).^2).*urJ + ...
((st.*kt2.*uiJ) ./ (kt2 + (uiJ).^2).^2).*uiJ - (1./2) .* ...
((st ./ (1 + (uiJ ./ kt).^2)) + ( st ./ (1 + (UI ./ kt).^2))));

FrUiKK = (- ht ./ (h2.^2)) .* ((ct.*kt2.*uiKK) ./ (kt2 + (uiKK).^2) ...
.^2).*UR    + (ht ./ (h2.^2)) .* ((st.*kt2.*uiKK) ./ (kt2 + (uiKK).^2) ...
.^2).*UI - (ht ./ (h2.^2)) .* ((-(ct.*kt2.*uiKK) ./ (kt2 + (uiKK) ...
.^2).^2).*urKK  + ((st.*kt2.*uiKK) ./ (kt2 + (uiKK).^2).^2).*uiKK - ...
(1./2) .*   ((st ./ (1 + (uiKK ./ kt).^2)) + ...
( st ./ (1 + (UI ./ kt).^2))));
```

```
152
153     FrUiK = (— ht ./ (h2.^2)) .* ((ct.*kt2.*uiK) ./ ...
154     (kt2 + (uiK).^2).^2).*UR + (ht ./ (h2.^2)) .* ((st.*kt2.*uiK) ./ ...
155     (kt2 + (uiK).^2).^2).*UI — (ht ./ (h2.^2)) .* ((—(ct.*kt2 .*uiK) ./ ...
156     (kt2 + (uiK).^2).^2).*urK + ((st.*kt2.*uiK) ./ (kt2 + (uiK).^2).^2) ...
157     .*uiK — (1./2) .*   ((st ./ (1 + (uiK ./ kt).^2)) + ...
158     ( st ./ (1 + (UI ./ kt).^2))));
159
160     FrUr = 1 + (ht ./ (2.*h1.^2)) .* ((ct ./(1 + (uiJJ ./ kt).^2)) + ...
161     (2.*ct./(1 + (UI ./ kt).^2)) + (ct./(1 + (uiJ./ kt).^2))) ...
162     + (ht./(2.*h2.^2)) .* ((ct./(1 + (uiKK ./ kt).^2)) + (2.*ct./(1 + ...
163     (UI./ kt).^2)) + (ct./(1 + (uiK ./ kt).^2)));
164
165     FrUrJJ = (—ht ./ (2.*h1.^2)) .* (ct./(1 + (uiJJ./ kt).^2) + ...
166     (ct./(1 + (UI./ kt).^2)));
167
168     FrUrJ = (—ht ./ (2.*h1.^2)) .* (ct./(1 + (uiJ./ kt).^2) + ...
169     (ct./(1 + (UI./ kt).^2)));
170
171     FrUrKK = (—ht ./ (2.*h2.^2)) .* (ct./(1 + (uiKK./ kt).^2) + ...
172     (ct./(1 + (UI./ kt).^2)));
173
174     FrUrK = (—ht ./ (2.*h2.^2)) .* (ct./(1 + (uiK./ kt).^2) + ...
175     (ct./(1 + (UI./ kt).^2)));
176
177     FiUi =  (ht./(h1.^2)) .* (( —(2.*ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2) + ...
178     ht./(h2.^2) .* ( —(2.*ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2)) .* UI ...
179     + (1+(ht./(2.*h1.^2)) .* ((ct ./ (1+ (uiJJ).^2 ./ kt2)) + (2 .* ct ...
180     ./ (1+ (UI).^2 ./ kt2)) + (ct ./ (1+ (uiJ).^2 ./ kt2))) + (ht./(2.* ...
181     h2.^2)) .*  ((ct ./ (1+ (uiKK).^2 ./ kt2)) + (2 .* ct ./ (1+ (UI).^2 ...
182     ./ kt2)) + (ct ./ (1+ (uiK).^2 ./ kt2)))) + ((ht./(h1.^2)) .* ((—(2 ...
183     .*st.*kt2.*UI) ./ (kt2 + (UI).^2).^2)) + (ht./(h2.^2)) .* ( —(2.* ...
184     st.*kt2.*UI) ./ ((kt2 + (UI).^2).^2))) .* UR + (ht./(h1.^2)) .* ...
185     ((((ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2) .*uiJJ + ((st.*kt2.*UI) ./ ...
186     (kt2 + (UI).^2).^2).*urJJ)+ (((ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2).* ...
187     uiJ + ((st.*kt2.*UI) ./ (kt2 + (UI).^2).^2) .*urJ)) + (ht./(h2.^2)) ...
188     .* ((((ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2) .*uiKK + ((st.*kt2.*UI) ./ ...
189     (kt2 + (UI).^2).^2).*urKK) + (((ct.*kt2.*UI) ./ (kt2 + (UI).^2).^2).* ...
190     uiK + ((st.*kt2.*UI) ./ (kt2 + (UI).^2).^2).*urK));
```

```
191
192     FiUiJJ = (ht./(h1.^2)) .* (-(ct.*kt2.*uiJJ) ./ (kt2 + (uiJJ).^2).^2) ...
193     .*UI + (ht./(h1.^2)) .* (-(st.*kt2.*uiJJ) ./ (kt2 + (uiJJ).^2).^2).*UR ...
194     - (ht ./(2.*h1.^2)) .* (((-ct.*kt2.*2.*uiJJ) ./ (kt2 + (uiJJ).^2) ...
195     .^2).*uiJJ  + ((ct./ (1 + (uiJJ ./ kt).^2)) + ( ct ./ (1 + (UI./ kt) ...
196     .^2)))- ((st.*kt2.*2.*uiJJ) ./ (kt2 + (uiJJ).^2).^2).*urJJ);
197
198     FiUiJ = (ht./(h1.^2)) .* (-(ct.*kt2.*uiJ) ./ (kt2 + (uiJ).^2).^2).*UI ...
199     + (ht./(h1.^2)) .* (-(st.*kt2.*uiJ) ./ (kt2 + (uiJ).^2).^2).*UR ...
200     - (ht ./(2.*h1.^2)) .* (((-ct.*kt2.*2.*uiJ) ./ (kt2 + (uiJ).^2).^2) ...
201     .*uiJ   + ((-ct./ (1 + (uiJ ./ kt).^2)) + ( ct ./ ...
202     (1 + (UI./ kt).^2)))-   ((st.*kt2.*uiJ) ./ (kt2 + (uiJ).^2).^2).*urJ);
203
204     FiUiKK = (ht./(h2.^2)) .* (-(ct.*kt2.*uiKK) ./ (kt2 + (uiKK).^2).^2) ...
205     .*UI + (ht./(h2.^2)) .* (-(st.*kt2.*uiKK) ./ (kt2 + (uiKK).^2).^2) ...
206     .*UR - (ht ./(2.*h2.^2)) .* (((-ct.*kt2.*2.*uiKK) ./ (kt2 + (uiKK).^2) ...
207     .^2).*uiKK  + ((ct./ (1 + (uiKK ./ kt).^2)) + ( ct ./ (1 + (UI./ kt) ...
208     .^2)))- ((st.*kt2.*2.*uiKK) ./ (kt2 + (uiKK).^2).^2).*urKK );
209
210     FiUiK = (ht./(h2.^2)) .* (-(ct.*kt2.*uiK) ./ (kt2 + (uiK).^2).^2).*UI ...
211     + (ht./(h2.^2)) .* (-(st.*kt2.*uiK) ./ (kt2 + (uiK).^2).^2).*UR ...
212     - (ht ./(2.*h2.^2)) .* (((-ct.*kt2.*2.*uiK) ./ (kt2 + (uiK).^2).^2) ...
213      .*uiK + ((ct./ (1 + (uiK ./ kt).^2)) + ( ct ./ (1 + (UI./ kt).^2))) - ...
214      ((st.*kt2.*2.*uiK) ./ (kt2 + (uiK).^2).^2).*urK );
215
216     FiUr = (ht ./ (2.*h1.^2)) .* ((st ./(1 + (uiJJ ./ kt).^2)) + ...
217     (2.*st./(1 + (UI ./ kt).^2)) + (st./(1 + (uiJ./ kt).^2))) ...
218     + (ht./(2.*h2.^2)) .* ((st./(1 + (uiKK ./ kt).^2)) + ...
219     (2.*st./(1 + (UI./ kt).^2) + (st./(1 + (uiK ./ kt).^2))));
220
221     FiUrJJ = (-ht ./ (2.*h1.^2)) .* (st./(1 + (uiJJ./ kt).^2) + ...
222     (st./(1 + (UI./ kt).^2)));
223
224     FiUrJ = (-ht ./ (2.*h1.^2)) .* (st./(1 + (uiJ./ kt).^2) + ...
225     (st./(1 + (UI./ kt).^2)));
226
227     FiUrKK = (-ht ./ (2.*h2.^2)) .* (st./(1 + (uiKK./ kt).^2) + ...
228     (st./(1 + (UI./ kt).^2)));
229
```

```
230     FiUrK = (-ht ./ (2.*h2.^2)) .* (st./(1 + (uiK./ kt).^2) + ...
231     (st./(1 + (UI./ kt).^2)));
232
233     % Monta o vetor F(u)
234     vFR = reshape(FR,[nPontos,1]);
235     vFI = reshape(FI,[nPontos,1]);
236     VetorFU = [vFR;vFI];
237
238     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
239     %FrUr diagonal principal
240     aux = reshape(FrUr.',[nPontos,1]);
241     for iter = 1 : nPontos
242        BL1(iter,iter) = aux(iter);
243     end
244     %FrUrK
245     aux = reshape(FrUrK.',[nPontos,1]);
246     for iter = 2 : nPontos
247        BL1(iter,iter-1) = aux(iter);
248     end
249     %FrUrKK
250     aux = reshape(FrUrKK.',[nPontos,1]);
251     for iter = 1 : nPontos-1
252        BL1(iter,iter+1) = aux(iter);
253     end
254     %FrUrJJ
255     aux = reshape(FrUrJJ.',[nPontos,1]);
256     for iter = 1 : nPontos-n1
257        BL1(iter,iter+n1) = aux(iter);
258     end
259     %FrUrJ
260     aux = reshape(FrUrJ.',[nPontos,1]);
261     for iter = 1+n1 : nPontos
262        BL1(iter,iter-n1) = aux(iter);
263     end
264     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
265     %FrUi diagonal principal
266     aux = reshape(FrUi.',[nPontos,1]);
267     for iter = 1 : nPontos
268        BL2(iter,iter) = aux(iter);
```

```
269     end
270     %FrUiK
271     aux = reshape(FrUiK.',[nPontos,1]);
272     for iter = 2:nPontos
273        BL2(iter,iter-1) = aux(iter);
274     end
275     %FrUiKK
276     aux = reshape(FrUiKK.',[nPontos,1]);
277     for iter = 1 : nPontos-1
278        BL2(iter,iter+1) = aux(iter);
279     end
280     %FrUiJJ
281     aux = reshape(FrUiJJ.',[nPontos,1]);
282     for iter = 1 : nPontos- n1
283        BL2(iter,iter+n1) = aux(iter);
284     end
285     %FrUiJ
286     aux = reshape(FrUiJ.',[nPontos,1]);
287     for iter = 1+n1 : nPontos
288        BL2(iter,iter-n1) = aux(iter);
289     end
290     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
291     %FiUr diagonal principal
292     aux = reshape(FiUr.',[nPontos,1]);
293     for iter = 1 : nPontos
294        BL3(iter,iter) = aux(iter);
295     end
296     %FiUrK
297     aux = reshape(FiUrK.',[nPontos,1]);
298     for iter = 2 : nPontos
299        BL3(iter,iter-1) = aux(iter);
300     end
301     %FiUrKK
302     aux = reshape(FiUrKK.',[nPontos,1]);
303     for iter = 1 : nPontos-1
304        BL3(iter,iter+1) = aux(iter);
305     end
306     %FiUrJJ
307     aux = reshape(FiUrJJ.',[nPontos,1]);
```

```
308     for iter = 1 : nPontos-n1
309        BL3(iter,iter+n1) = aux(iter);
310     end
311     %FiUrJ
312     aux = reshape(FiUrJ.',[nPontos,1]);
313     for iter = 1+n1 : nPontos
314        BL3(iter,iter-n1) = aux(iter);
315     end
316     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
317     %FiUi diagonal principal
318     aux = reshape(FiUi.',[1,nPontos]);
319     for iter = 1 : nPontos
320        BL4(iter,iter) = aux(1,iter);
321     end
322     %FiUiK
323     aux = reshape(FiUiK.',[nPontos,1]);
324     for iter = 2 : nPontos
325        BL4(iter,iter-1) = aux(iter);
326     end
327     %FiUiKK
328     aux = reshape(FiUiKK.',[nPontos,1]);
329     for iter = 1 : nPontos-1
330        BL4(iter,iter+1) = aux(iter);
331     end
332     %FiUiJJ
333     aux = reshape(FiUiJJ.',[nPontos,1]);
334     for iter = 1 : nPontos-n1
335        BL4(iter,iter+n1) = aux(iter);
336     end
337     %FiUiJ
338     aux = reshape(FiUiJ.',[nPontos,1]);
339     for iter = 1+n1 : nPontos
340        BL4(iter,iter-n1) = aux(iter);
341     end
342
343     MatrizFl = [BL1,BL2;BL3,BL4]; % monta a Matriz Fl a partir dos 4 blocos.
344
345     Unovo = MatrizFl\VetorFU; % atribui UR para U, x0
346
```

```
347     L2dif = sum(Unovo(:).^2)/2/nPontos; % recalcula o criterio de paraga
348
349     UR = UR - reshape((Unovo(1:nPontos)).', [n1,n2]);
350     UI = UI - reshape((Unovo(nPontos+1:end)).', [n1,n2]);
351
352     % Exibe uma mensagem que parou pela iteracao e nao pela L2dif
353     if iteracao == maxIter
354        fprintf('\nExecucao interrompida pois numero de iteracoes do Metodo ...
355        de Newton e igual ao numero maximo de iteracoes definido: %d',maxIter);
356     end
357
358 endwhile % loop da iteracao de newton
359
360 fprintf('\nPasso numero: %d - Iteracoes: %d', passo, iteracao)
361
362 URN = UR;
363 UIN = UI;
364 U(:,:,passo+1) = URN+i*UIN;
365
366 end % loop do metodo implicito
367
368 t = toc;
369 fprintf('\nTempo de execucao: %s \n',datestr(datenum(0,0,0,0,0,t),'HH:MM:SS'))
```