MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2018/04.26.15.43-TDI

MÉTODOS IMEX PARA INTEGRAÇÃO TEMPORAL DA EQUAÇÃO DE BURGERS

Antonio Maurício Zarzur

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Stephan Stephany, e Saulo Ribeiro de Freitas, aprovada em 27 de abril de 2018.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3R25SCH>

INPE
São José dos Campos
2018

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2018/04.26.15.43-TDI

MÉTODOS IMEX PARA INTEGRAÇÃO TEMPORAL DA EQUAÇÃO DE BURGERS

Antonio Maurício Zarzur

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Stephan Stephany, e Saulo Ribeiro de Freitas, aprovada em 27 de abril de 2018.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/3R25SCH>

INPE
São José dos Campos
2018

Dados Internacionais de Catalogação na Publicação (CIP)

Zarzur, Antonio Maurício.

Za19m Métodos IMEX para integração temporal da Equação de Burgers / Antonio Maurício Zarzur. – São José dos Campos : INPE, 2018.

xx + 68 p. ; (sid.inpe.br/mtc-m21c/2018/04.26.15.43-TDI)

Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2018.

Orientadores : Drs. Stephan Stephany, e Saulo Ribeiro de Freitas.

1. Diferenças finitas. 2. Equações diferenciais parciais. 3. Esquemas IMEX. 4. Integração temporal. 5. Modelagem numérica. I.Título.

CDU 517.91

Aluno (a): *Antonio Mauricio Zarzur*
Título: **"MÉTODOS IMEX PARA INTEGRAÇÃO TEMPORAL DA EQUAÇÃO DE BURGERS".**

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Mestre* em *Computação Aplicada*

Dr. Haroldo Fraga de Campos Velho

Presidente / INPE / São José dos Campos - SP

( ) Participação por Video - Conferência

Dr. Stephan Stephany

Orientador(a) / INPE / SJCampos - SP

( ) Participação por Video - Conferência

Dr. Saulo Ribeiro de Freitas

Orientador(a) / INPE / São José dos Campos - SP

(X) Participação por Video - Conferência

Dr. Reinaldo Roberto Rosa

Membro da Banca / INPE / SJCampos - SP

( ) Participação por Video - Conferência

Dr. Pedro Leite da Silva Dias

Convidado(a) / USP / São Paulo - SP

(X) Participação por Video - Conferência

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 27 de abril de 2018

"When day becomes dusk
Don't let that stop us
On our way to something more uncertain".

Casey Crescenzo
em *"All Is As All Should Be"*, 2017

AGRADECIMENTOS

Ao Dr. Haroldo Fraga de Campos Velho, não apenas pela colaboração e pelo incentivo, mas também por sua infinita disposição para ajudar na melhoria deste trabalho.

Ao Dr. Saulo Ribeiro de Freitas, pela confiança e dedicação demonstrada desde meus primeiros dias no CPTEC, oferecendo-me a oportunidade de expandir os conhecimentos adquiridos sob sua chefia e orientação.

Ao Dr. Stephan Stephany, por toda a ajuda, orientação e paciência ao longo da realização deste trabalho.

Aos amigos Antonio de Oliveira, Cherise Wheeler, Diana Cruz, Diego Monteiro, Elie Simard, Felipe Zanquetta, Hélio Baroni, Jeff Biddick, Mauricio Sousa, Rachel Abdala e Silvia Ferreira pelos momentos de descontração e pelo apoio nos momentos mais difíceis.

Aos amigos e ex-colegas do CPTEC/INPE, em especial Rodrigo Braz, Valter Oliveira e Julliana Freire, por proporcionarem um ambiente de trabalho tanto agradável quanto propício a novos questionamentos.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro durante a realização deste trabalho.

RESUMO

Simulações computacionais baseiam-se em modelos matemáticos desenvolvidos para certas classes de fenômenos. A solução computacional de equações diferenciais parciais requer a escolha de um método de integração temporal capaz de simular, de forma estável, a evolução do problema. Há métodos mais adequados para determinadas classes de fenômenos, não existindo um método geral que sirva adequadamente para todos os fenômenos. Deve-se levar em conta a acurácia e a estabilidade do método adotado, bem como seu desempenho computacional. De forma geral, os métodos de integração temporal são classificados como implícitos ou explícitos. Cada família apresenta vantagens e desvantagens na solução de determinadas classes de problemas. Uma abordagem mais recente, denominada IMEX, visa combinar as vantagens de cada estratégia para solucionar equações com escalas de tempo variáveis, de forma que os termos rápidos são resolvidos implicitamente e os mais lentos são resolvidos explicitamente. O resultado é uma combinação de diferentes esquemas que otimiza o tempo de processamento ao evitar passos de tempo desnecessariamente pequenos para os termos rápidos. Este trabalho propõe a aplicação dessa abordagem na solução da equação de Burgers viscosa, objetivando realizar um estudo de caso, analisando sua acurácia e desempenho computacional. A equação de Burgers é uma das equações fundamentais da dinâmica de fluidos e serve como uma simplificação das equações de Navier-Stokes sem a presença da equação de continuidade e dos gradientes de pressão, possuindo assim diversas aplicações práticas. Sua solução exata é conhecida, o que permite a comparação da acurácia dos métodos IMEX propostos com os tradicionais esquemas implícitos ou explícitos. Experimentos numéricos demonstram que os métodos propostos produzem soluções com o mesmo grau de acurácia dos métodos tradicionais, ao mesmo tempo em que estendem as condições de estabilidade além dos limites de métodos puramente explícitos.

Palavras-chave: Diferenças finitas. Equações diferenciais parciais. Esquemas IMEX. Integração temporal. Modelagem numérica.

IMEX METHODS FOR TIME INTEGRATION OF BURGERS' EQUATION

ABSTRACT

Computational simulations are based on mathematical models developed for certain phenomena. The numerical solution of partial differential equations requires the choice of a method for time integration capable of stably simulating the evolution of a problem. There are methods that are more suitable to certain classes of phenomena, and therefore no single, general method can be applied to every problem. Both accuracy and stability of the chosen method must be taken into account, as well as its computational efficiency. Generally speaking, time integration schemes are categorized as either implicit or explicit. Each of these broad families presents pros and cons when solving certain classes of problems. A more modern approach, called IMEX, seeks to combine the advantages of each strategy to solve equations containing both fast and slow time-scales in a way that the slow terms can be solved explicitly, while the slow terms are solved implicitly. This results in a combination of different schemes with the goal of optimizing processing time by avoiding unnecessarily small time steps for the fast terms. This dissertation utilizes this approach in solving the viscous Burgers' equation as a test case for such methods, analyzing their accuracy and computational performance. Burgers' equation is one of the fundamental equations in fluid dynamics which essentially simplifies the Navier-Stokes equations by removing the pressure gradient terms and continuity equation, thus serving diverse practical applications. Because its exact solution is known, comparisons can be drawn between the accuracy of the proposed IMEX schemes and that of more traditional implicit or explicit schemes. Numerical experiments are performed to demonstrate their ability to simulate the problem with the same order of accuracy achieved by traditional means, while extending the stability conditions beyond the reach of purely explicit schemes.

Keywords: Finite Differences. IMEX Schemes. Partial Differential Equations. Time Integration. Numerical Modeling.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

AB	–	Adams-Bashforth
AM	–	Adams-Moulton
BRAMS	–	Brazilian developments on the Regional Atmospheric Modeling System
CFL	–	Courant-Friedrichs-Lewy
COSMO	–	Consortium for Small-scale Modeling
CPTEC	–	Centro de Previsão de Tempo e Estudos Climáticos
EDP	–	Equação Diferencial Parcial
FTCS	–	Forward-Time, Central-Space
IMEX	–	Implicit-Explicit
INPE	–	Instituto Nacional de Pesquisas Espaciais
LAPACK	–	Linear Algebra Package
RK	–	Runge-Kutta
WRF	–	Weather Research and Forecasting Model

SUMÁRIO

Pág.

1 INTRODUÇÃO

Muitos dos fenômenos físicos mais importantes na natureza podem ser modelados por equações diferencias parciais. Em princípio, é possível simular por métodos numéricos aproximados qualquer fenômeno cujas equações governantes sejam conhecidas. No caso de equações de evolução, pode-se gerar previsões para seus estados futuros por meio da integração temporal dos sistemas dinâmicos resultantes da discretização espacial dessas equações. Para tal, é necessário partir de um estado inicial conhecido (doravante denominado "condição inicial") e conhecer o comportamento das fronteiras espaciais do fenômeno (doravante denominado "condição de contorno").

Os métodos tradicionais de integração temporal são normalmente classificados em duas famílias: explícitos e implícitos. Métodos explícitos são facilmente implementáveis, já que utilizam apenas informações previamente conhecidas para calcular o passo de tempo seguinte, mas sua estabilidade é condicionada por uma relação entre a discretização temporal e a espacial[1]. Consequentemente, os passos de tempo necessários para a evolução de um problema com variação rápida podem ser restritos a valores demasiadamente pequenos. Esquemas implícitos, por sua vez, podem levar a uma abordagem incondicionalmente estável – sem restrições entre as discretizações – mas por outro lado podem resultar em sistemas de equações algébricas não lineares cuja solução é complexa.

Uma abordagem alternativa, denominada IMEX (*IMplicit-EXplicit*), combina essas duas famílias em um esquema composto, desenvolvido para resolver problemas que contenham tanto escalas de tempo rápidas quanto lentas. Essa classe de problemas foi estudada detalhadamente nos trabalhos de Ascher et al. (1995) e Ascher et al. (1997), e seus sistemas de equações normalmente assumem a forma:

$$u_t = f(u) + \nu g(u) \tag{1.1}$$

onde subscritos denotam derivadas e ν é um parâmetro não negativo. Na Equação 1.1, os termos coletados em $f(u)$ apresentam uma escala de tempo lenta, e

[1] Um estudo amplamente conhecido é a análise númérica da equação de evolução de convecção: $\partial\varphi/\partial t + c\,\partial\varphi/\partial x$, sendo c a velocidade da onda, onde a condição de estabilidade tornou-se conhecida como *condição de Courant-Friedrichs-Lewy*, ou CFL (COURANT et al., 1928). A convergência dos métodos numéricos na solução de EDP está ligada ao *teorema da equivalência de Lax* (LAX; RICHTMYER, 1956): para problemas de valor inical de EDP lineares, na aplicação de métodos numéricos consistentes e estáveis, a solução computacional se torna mais próxima da solução exata quando as discretizações tornam-se cada vez menores, i.e., a solução numérica *converge* para a solução exata.

portanto não costumam causar instabilidades; no entanto, como são possivelmente não lineares, a implementação de esquemas implícitos encontra desafios relacionados ao fraco desempenho dos *solvers* iterativos necessários ou da complexidade da matrix Jacobiana associada ao sistema. Faria sentido, então, resolve-lo de forma explícita. Os termos coletados em $g(u)$, no entanto, são resolvidos em uma escala de tempo rápida, de forma que uma solução explícita pode requerer passos de tempo demasiadamente pequenos. Sendo normalmente lineares, tais termos podem ser resolvidos implicitamente sem complicações adicionais.

Os esquemas IMEX são propostos então a partir dessa exata estratégia – a combinação de um esquema implícito para os termos rápidos em $g(u)$ com um esquema explícito para os termos lentos em $f(u)$ – com foco na estabilidade e acurácia do modelo combinado (WELLER et al., 2013). O esquema resultante combina dois métodos previamente sem relação com o intuito de otimizar o tempo de processamento, evitando a geração de sistemas não lineares complexos e a utilização de passos de tempo pequenos. Devido à presença de um esquema explícito, métodos IMEX ainda são métodos *condicionalmente estáveis*, mas métodos desse tipo possuem restrições menos severas com relação ao tamanho do passo de tempo devido ao tratamento implícito das escalas rápidas. Se uma condição de estabilidade for obtida tal que a redução no número de passos necessários para completar a simulação seja suficiente para compensar o custo adicional na computação de cada passo individual, é possível aumentar o desempenho geral do algoritmo.

1.1 Motivação

Muitas aplicações relevantes no campo da mecânica de fluidos computacional utilizam métodos explícitos para integração temporal de suas simulações. Modelos de previsão numérica do tempo como o BRAMS (FREITAS et al., 2017), COSMO (BALDAUF, 2008) e WRF (WICKER; SKAMAROCK, 1998) (WICKER; SKAMAROC, 2002) utilizam integradores temporais Runge-Kutta e *leapfrog*. Estudos de fluxo e transferência de calor em nanofluidos (NOGHREHABADI et al., 2014) e suas camadas limite (GHALAMBAZ et al., 2015) também empregam esquemas Runge-Kutta. No entanto, com o aumento das resoluções espaciais, o custo computacional de tais métodos pode ser tornar não competitivo.

O modelo BRAMS, desenvolvido no Centro de Previsão de Tempo e Estudos Climáticos (CPTEC) do Instituto Nacional de Pesquisas Espaciais (INPE), é capaz de realizar a previsão numérica do tempo em toda a América do Sul com uma resolução horizontal de cinco por cinco quilômetros. A grade resultante em uma resolução

desse porte possui 1360×1489 pontos, com 55 níveis verticais, necessitando de matrizes com um número de elementos na ordem de 10^8 (CPTEC, 2016). Para que os cálculos sejam possíveis utilizando a opção de integração *leapfrog*, é necessário um passo de tempo de apenas 15 segundos, demandando um grande tempo de processamento mesmo com a utilização de 9600 processadores do supercomputador Tupã do CPTEC-INPE.

Durran e Blossey (2012) propõem uma nova família IMEX baseados nos métodos de Adams, que apresentam propriedas de estabilidade atrativas para certos problemas da mecânica de fluidos, incluindo possíveis aplicações relacionadas à dinâmica da atmosfera. A formulação geral da família permite diferentes combinações de esquemas implícitos/explícitos com base em um conjunto de parâmetros numéricos, produzindo assim múltiplas opções para a solução de um dado problema.

Esta dissertação propõe uma série de experimentos numéricos com o objetivo de verificar a eficiência e acurácia dos métodos de Durran e Blossey na solução de problemas típicos da mecânica de fluidos computacional. Com esse fim, propõe-se a solução numérica da equação de Burgers – uma das equações fundamentais da área – por métodos IMEX. Por possuir solução analítica conhecida, a equação de Burgers é amplamente utilizada como plataforma de testes para algoritmos numéricos. Dessa forma, casos de teste já validados na literatura podem ser escolhidos para verificar a implementação computacional dos novos métodos e sua comparar sua acurácia e eficiência a uma solução explícita e uma implícita (igualmente validades).

1.2 Publicações derivadas desta dissertação

Quatro publicações foram geradas com base na pesquisa desenvolvida nesta dissertação, incluindo dois resumos e dois artigos completos publicados em congressos.

Estágios iniciais da pesquisa foram publicados na forma de um resumo na quarta edição da *Conference of Computational Interdisciplinary Science*, realizada em 2016 no INPE de São José dos Campos (ZARZUR et al., 2016). Foram propostas implementações explícitas e implícitas, além de um esquema IMEX clássico baseado na combinação entre os métodos de Crank-Nicolson e Adams-Bashforth (CNAB).

A evolução desse trabalho foi apresentada no XXXVII Congresso Nacional de Matemática Aplicada e Computacional, em 2017, também em forma de resumo (ZARZUR et al., 2018). Foram apresentados os resultados da primeira implementação do esquema CNAB, que ainda continha *bugs* na solução do sistema linear resultante da

discretização IMEX no momento da submissão, mas cujos resultados corrigidos foram apresentados no congresso.

Uma versão mais avançada da pesquisa foi apresentada no *Ibero-Latin American Congress on Computational Methods in Engineering* e publicada nos anais do evento (ZARZUR et al., 2017). Neste trabalho, um primeiro método baseado em Durran e Blossey (AI2*-AB3) e um método similar (AM2*-AB3) foram apresentados. Embora ainda não otimizada para desempenho computacional, as implementações foram comparadas com o método de Crank-Nicolson com relação à sua performance, apresentando ganhos significativos.

Os resultados finais da implementação dos métodos de Durran e Blossey foram apresentados na terceira edição da Thermal and Fluids Engineering Conference (TFEC), em Fort Lauderdale, e publicados nos anais do evento (ZARZUR et al., 2018). Nesse trabalho, os três métodos apresentados nesta dissertação foram comparados aos métodos explícito (FTCS) e implícito (Crank-Nicolson) também estudados com destaque dado à acurácia dos métodos testados. A implementação utilizada na elaboração do artigo era ainda ingênua e desprovida de otimizações de código, e desta forma foi novamente comparada apenas à solução implícita com relação à sua performance computacional.

1.3 Organização do texto

Os capítulos restantes desta dissertação estão organizados da seguinte maneira:

- Capítulo 2: Aborda o método das diferenças finitas para solução de EDPs no âmbito da mecânica de fluidos computacional. São mencionadas as diferenças entre esquemas explícitos e implícitos, e apresentados os esquemas do tipo IMEX. Os esquemas desenvolvidos por Durran e Blossey são detalhados.

- Capítulo 3: A equação de Burgers é apresentada como um modelo simplificado para testes de algoritmos numéricos. São detalhados esquemas de integração temporal explícita, implícita e IMEX para equação de Burgers nos casos unidimensional e bidimensional. São apresentadas as equações discretizadas.

- Capítulo 4: Descrição dos experimentos numéricos para validação das implementações propostas no Capítulo 3. São apresentadas as soluções analíticas utilizadas como condições iniciais e de contorno para as simulações.

Resultados numéricos são comparados à solução analítica e uma análise comparativa da eficiência computacional de cada método é detalhada.

- Capítulo 5: São apresentadas as conclusões, comentários finais e sugestões de trabalhos futuros.

2 TÉCNICAS PARA INTEGRAÇÃO TEMPORAL DE EDP

A mecânica de fluidos computacional é uma metodologia interdisciplinar para solução de sistemas físicos relacionados a fluxo de fluidos, transferência de calor e outros fenômenos relacionados por meio de simulações numéricas, aproveitando-se dos avanços constantes da tecnologia computacional. Tais sistemas, conforme mencionado no Capítulo 1, são governados por equações diferenciais parciais.

A solução analítica dessas equações resulta em expressões de forma fechada, que fornecem a variação das variáveis dependentes de forma contínua por todo o domínio do problema (ANDERSON JUNIOR, 1996). Soluções numéricas, no entanto, são limitadas no sentido de que podem fornecer valores apenas em pontos discretos espaçados ao longo do domínio. Estes são os chamados pontos de grade. De forma equivalente, na dimensão temporal, o aspecto contínuo das equações é substituído por pequenos incrementos discretos ao longo do tempo. Simulações de fenômenos físicos por este método partem então de um estado inicial conhecido e integram as equações discretizadas ao longo do tempo, gerando novos valores para a solução em cada passo de tempo discreto (NIELSEN et al., 2014).

O método das diferenças finitas tem por objetivo substituir os termos derivativos espaciais em uma EDP por expressões que aproximem seus valores nos pontos de grade, resultando em uma equação algébrica que pode ser resolvida computacionalmente. Outros métodos existem para resolver numericamente sistemas de EDPs, como elementos finitos, volumes finitos e métodos espectrais (WANG; RUUTH, 2008). No entanto, o foco deste trabalho é o método das diferenças finitas, amplamente utilizado em modelos atmosféricos e demais aplicações da mecânica de fluidos computacional.

2.1 Representação de derivadas em diferenças finitas

As representações das derivadas em diferenças finitas são construídas com base na expansão da função em uma série de Taylor. Por questão de simplicidade, considere inicialmente o problema:

$$\frac{du}{dx} = f(u, x), \quad u = u(x) \tag{2.1}$$

onde variável independente x é definida em pontos discretos com espaçamento Δx, e a notação u_i representa o valor de u no ponto $x = i\Delta x,\ (i \in \mathbb{Z})$.

O valor de u_{i+1} pode ser expresso na forma de uma série de Taylor como:

$$u_{i+1} = u_i + \Delta x \left(\frac{du}{dx}\right)_i + \frac{(\Delta x)^2}{2!}\left(\frac{d^2u}{dx^2}\right)_i + \frac{(\Delta x)^3}{3!}\left(\frac{d^3u}{dx^3}\right)_i + ... \tag{2.2}$$

De acordo com Wendt (2009), a equação 2.2 é uma expressão exata para u_{i+1} se:

a) o número de termos é infinito e a série converge,

b) e/ou $\Delta x \to 0$.

Obviamente, é inviável utilizar séries com infinitos termos em computação. Sendo assim, a série de Taylor deve ser truncada em algum ponto, gerando um erro na aproximação (chamado erro de truncamento) igual à somatória dos termos ignorados. A chamada ordem de acurácia da aproximação é determinada pelo número de termos da série que não são ignorados. Por exemplo, a expressão para u_{i+1}:

$$u_{i+1} \approx u_i + \Delta x \left(\frac{du}{dx}\right)_i \tag{2.3}$$

tem acurácia de primeira ordem, pois os termos de ordem $(\Delta x)^2$ e superiores são ignorados. Resolvendo a equação 2.3 para a derivada, obtemos a diferença finita progressiva (*forward difference*) de primeira ordem para u_{i+1}:

$$\left(\frac{du}{dx}\right)_i = \frac{u_{i+1} - u_i}{\Delta x} + O(\Delta x) \tag{2.4}$$

onde $O(\Delta x)$ é o erro de truncamento de ordem (Δx). De forma equivalente, resolvendo para u_{i-1}, temos a série de Taylor:

$$u_{i-1} = u_i - \Delta x \left(\frac{du}{dx}\right)_i + \frac{(-\Delta x)^2}{2!}\left(\frac{d^2u}{dx^2}\right)_i + \frac{(-\Delta x)^3}{3!}\left(\frac{d^3u}{dx^3}\right)_i + ... \tag{2.5}$$

que truncada fornece a diferença finita regressiva (*backward difference*) de primeira ordem:

$$\left(\frac{du}{dx}\right)_i = \frac{u_i - u_{i-1}}{\Delta x} + O(\Delta x) \tag{2.6}$$

Subtraindo a equação 2.5 de 2.2, chega-se a:

$$u_{i+1} - u_{i-1} = 2\Delta x \left(\frac{du}{dx}\right)_i + \frac{2(\Delta x)^3}{3!}\left(\frac{d^3u}{dx^3}\right)_i + ... \tag{2.7}$$

que fornece a diferença finita central de segunda ordem:

$$\left(\frac{du}{dx}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x)^2 \tag{2.8}$$

Muitas outras expressões de diferenças finitas podem ser obtidas utilizando a mesma metodologia, incluindo aproximações para derivadas de ordem mais alta. Na seção 2.3 serão discutidas algumas aproximações para derivadas espaciais de segunda ordem.

2.2 Integração temporal

A discretização temporal de uma aproximação por diferenças finitas envolve a integração ao longo do tempo da equação geral discretizada. De acordo com Mesinger e Arakawa (1976), os termos temporais em EDPs normalmente são discretizadas por esquemas relativamente simples, com acurácia de segunda ou até mesmo primeira ordem, pois a experiência mostra que esquemas construídos com ordens muito altas não costumam ser bem sucedidos na solução de EDPs.

Esta seção do trabalho preocupa-se apenas com o termo da derivada temporal, sendo ignorados os aspectos espaciais. De fato, considera-se o problema na forma:

$$\frac{du}{dt} = g(u,t), \quad u = u(t) \tag{2.9}$$

onde variável independente t (tempo) é definida em pontos discretos com espaçamento Δt, e a notação u^n representa o valor de u no tempo $t = n\Delta t,\ (n \in \mathbb{Z})$.

Esquemas de integração temporal podem ser construídos utilizando diferentes números de passos de tempo. Nas seções 2.2.1 e 2.2.2 serão discutidos esquemas gerados com dois e três níveis temporais, respectivamente.

2.2.1 Esquemas com dois níveis temporais

Esquemas com dois níveis consideram os valores da variável dependente em apenas dois passos de tempo (n e $n+1$) e são os únicos que podem ser utilizados no pri-

meiro passo de tempo, quando apenas a condição inicial u^0 é conhecida (MESINGER; ARAKAWA, 1976). A forma geral para a discretização temporal da equação 2.9 em dois níveis pode ser escrita como uma média ponderada entre os valores do presente (passo de tempo n) e do futuro (passo de tempo $n+1$), ou seja:

$$\frac{u^{n+1} - u^n}{\Delta t} = \alpha g^{n+1} + (1-\alpha) g^n \tag{2.10}$$

onde α é o fator peso e varia entre 0 e 1. Diversos esquemas podem ser derivados da equação 2.10 dependendo do valor escolhido para α. Esquemas completamente explícitos podem ser obtidos com $\alpha = 0$, resultando na forma:

$$u^{n+1} = u^n + \Delta t \cdot g^n \tag{2.11}$$

Com $\alpha = 1$, é obtido um esquema completamente implícito da forma:

$$u^{n+1} = u^n + \Delta t \cdot g^{n+1} \tag{2.12}$$

Outros esquemas implícitos podem ser criados com diferentes valores de α; entre eles, um esquema notável é o trapezoidal, obtido quando $\alpha = 0.5$, gerando uma média aritmética simples entre os dois níveis temporais:

$$u^{n+1} = u^n + \Delta t \cdot \frac{g^{n+1} + g^n}{2} \tag{2.13}$$

As diferenças entre esquemas explícitos e implícitos serão exploradas na seção 2.2.3.

2.2.2 Esquemas com três níveis temporais

Esquemas com três níveis visam tirar vantagem de uma informação adicional: o valor de u^{n-1}. Esse valor não está definido no primeiro passo de tempo, por isso esquemas deste tipo precisam ser inicializados com algum esquema de dois níveis para o primeiro passo; após inicializado, os passos seguintes são informados pelos dois anteriores.

Um dos mais clássicos exemplos desta metodologia é o esquema chamado *leapfrog*, obtido pelo uso de diferença central de segunda ordem no termo temporal da equa-

ção 2.9. A equação resultante tem a forma:

$$u^{n+1} = u^{n-1} + 2\Delta t \cdot g^n \tag{2.14}$$

Um outro exemplo clássico é o esquema de Adams-Bashforth. Mesinger e Arakawa (1976) afirmam que a versão deste esquema normalmente utilizada em ciências atmosféricas é, na verdade, uma versão simplificada do esquema de quarta ordem de Adams-Bashforth. A versão simplificada é obtida a partir da equação:

$$u^{n+1} = u^n + \int_{n\Delta t}^{(n+1)\Delta t} g(u,t)dt \tag{2.15}$$

quando g é aproximada por um valor obtido no centro de Δt com uma extrapolação linear baseada em valores de g^n e g^{n-1}. O procedimento resulta no esquema de segunda ordem:

$$u^{n+1} = u^n + \Delta t \left(\frac{3}{2}g^n - \frac{1}{2}g^{n-1}\right) \tag{2.16}$$

Assim como no caso dos esquemas de dois níveis, é possível gerar inúmeras outras aproximações que não serão discutidas no escopo deste trabalho.

2.2.3 Caracterização de esquemas implícitos e explícitos

Na discussão sobre os vários métodos de integração temporal da seção 2.2, as denominações *implícito* e *explícito* foram utilizadas para caracterizar diferentes esquemas numéricos, mas não foram detalhadas. Esta seção visa explorar as diferenças entre os dois métodos, tanto do ponto de vista matemático quando com relação à sua implementação computacional.

Esquemas explícitos visam calcular um estado desconhecido, u^{n+1}, utilizando apenas informações já conhecidas dos passos de tempo anteriores:

$$u^{n+1} = f(u^n, u^{n-1}, u^{n-2}, ...) \tag{2.17}$$

Sendo assim, cada elemento da grade pode ser calculado diretamente e de forma independente dos demais, levando a uma implementação simples. No entanto, esses esquemas estão restritos às condições de CFL para estabilidade numérica: dados valores para os incrementos espaciais (Δx, Δy, etc.), o valor de Δt deve ser menor que um determinado limite dependente do esquema escolhido. Em muitos casos, o valor de Δt precisa ser muito pequeno para manter a estabilidade, resultando em

um tempo de computação excessivo devido ao número de passos necessários para completar a integração. Ocorrências desse tipo são particularmente presentes em sistemas rígidos (*stiff*), definidos por Chapra e Canale (2010) como sistemas que envolvem tanto componentes de variação rápida quanto de variação lenta.

Esquemas implícitos não possuem restrições semelhantes, e portanto são mais eficientes na solução de sistemas rígidos (NIELSEN et al., 2014). No entanto, as equações para cada ponto no passo de tempo $n+1$ são escritas em função não só dos passos anteriores, mas também dos valores de seus vizinhos no passo futuro:

$$u^{n+1} = f(u^{n+1}, u^n, u^{n-1}, u^{n-2}, ...) \tag{2.18}$$

Sendo assim, equações para todos os pontos de grade – que não podem ser resolvidas individualmente – são geradas, resultando em um sistema de equações algébricas que deve ser solucionado para fornecer, simultaneamente, os valores de u^{n+1} em todo o domínio. Dependendo do problema, esses sistemas podem ser de natureza não linear, necessitando de um método iterativo a cada passo de tempo. Devido às manipulações de matrizes necessárias para resolver os sistemas, o tempo de computação de cada passo de tempo se torna muito maior do que o de um esquema implícito. Por outro lado, os esquemas implícitos não são possuem restrição com relação ao valor de Δt, podendo então ser integrados em passos de tempo muito maiores.

2.2.4 Esquemas IMEX

Esquemas IMEX, conforme descritos no Capítulo 1, são uma classe de métodos desenvolvidos para resolver equações que contenham tanto termos rígidos quanto não-rígidos. Trata-se de uma abordagem mais recente, que visa combinar a simplicidade dos métodos explícitos com a estabilidade dos métodos implícitos. A estratégia é utilizada desde a década de 1980, frequentemente em conjunto com métodos espectrais – como nos trabalhos de Kim e Moin (1985) e Canuto et al. (1987). No entanto, é o artigo clássico de Ascher et al. (1995) que formaliza a abordagem IMEX. Os autores propõem inúmeras melhorias que se tornaram referência obrigatória no uso moderno desses métodos em aplicações como dinâmica de *icebergs* (LEMIEUX et al., 2014), equações de onda (WELLER et al., 2013), transmissão de calor (KADIOGLU; KNOLL, 2011) e muitos outras.

A abordagem baseia-se na observação que muitos dos fenômenos físicos representados por EDPs, após serem convenientemente transformados em sistemas de equações diferenciais ordinárias por meio das discretizações espaciais aqui discutidas, frequen-

temente tomam a forma:

$$u_t = f(u) + \nu g(u) \tag{2.19}$$

onde subscritos denotam derivadas e ν é um parâmetro não negativo.

Na equação 2.19, $f(u)$ representa o conjunto dos termos não-rígidos, ou seja, resolvidos numa escala de tempo lenta. Em geral, esses termos são não lineares e ligados a processos convectivos, mas não causam instabilidades. A solução implícita de $f(u)$ é indesejável, seja devido à complexidade ou densidade de sua matriz Jacobiana associada, por ser incompatível com o *solver* que se deseja utilizar, ou mesmo por uma questão de dificuldade de implementação. Faria sentido, então, resolver tais termos de forma explícita.

Os termos coletados em $g(u)$, por sua vez, são normalmente rígidos; ou seja, apresentam escalas de tempo rápidas. Tratam-se de processos difusivos lineares que resultam em sistemas esparsos, que podem ser resolvidos eficientemente com técnicas iterativas. Devido à rápida variação de seus componentes, é preferível resolver estes termos implicitamente, evitando assim passos de tempos demasiadamente pequenos para satisfazer às condições de estabilidade numérica.

Esquemas IMEX foram desenvolvidos partindo do princípio de que faria sentido, em casos como o descrito pela equação 2.19, atribuir um esquema explícito à solução de $f(u)$ e um esquema implícito à de $g(u)$, resultando assim em um novo método combinado:

$$u^{n+1} = f(u^n, u^{n-1}, u^{n-2}, ...) + \nu g(u^{n+1}, u^n, u^{n-1}, u^{n-2}, ...) \tag{2.20}$$

A princípio não existem limitações para as combinações possíveis, e inúmeras famílias de métodos IMEX podem ser construídas com base nessa filosofia. Deve-se então prestar atenção particular à estabilidade e acurácia do esquema combinado (WELLER et al., 2013) em relação ao problema proposto.

Diversas combinações possíveis foram exploradas na literatura das últimas décadas. O trabalho clássico de Ascher et al. (1995) apresenta combinações entre vários esquemas clássicos, com a metodologia posteriormente estendida para incluir combinações de métodos Runge-Kutta implícitos e explícitos denominadas RK IMEX (ASCHER et al., 1997). No âmbito da mecânica de fluidos, Durran e Blossey (2012) propõem duas famílias de métodos IMEX *multistep* voltados à solução de problemas relacionados à dinâmica da atmosfera – especificamente, problemas do tipo onda-lenta-onda-rápida, em que o valor máximo para um passo de tempo estável é determinado pela veloci-

dade da onda mais rápida. Métodos *multistep* ou multinível (como por exemplo os métodos de três níveis descritos na Seção 2.2.2) utilizam múltiplos níveis temporais para calcular o passo de tempo futuro e serão tratados na Seção 2.2.5.

2.2.5 Esquemas IMEX *multistep* lineares

Esquemas *multistep* se aproveitam das informações de mais de um estado (ou passo de tempo) para calcular o passo seguinte da simulação. Uma estratégia de integração temporal é chamada de s-nível (ou s-*step*) quando utiliza um total de s níveis temporais conhecidos em sua formulação. Uma expressão geral e consistente com os trabalhos de Ascher et al. (1995) e Durran e Blossey (2012) para obter-se métodos IMEX *multistep* lineares com $M+1$ níveis é dada por:

$$\sum_{k=-M}^{1} \alpha_k u^{n+k} = \Delta t \left[\sum_{k=-M}^{0} \beta_k f(u^{n+k}) + \nu \sum_{k=-M}^{1} \lambda_k g(u^{n+k}) \right] \tag{2.21}$$

Os parâmetros α, β e λ na Equação 2.21 devem ser escolhidos cuidadosamente, resultando na combinação dos esquemas explícito e implícito desejados.

Neste trabalho, foram escolhidos os esquemas baseados em métodos de Adams para os quais $\alpha_1 = 1$, $\alpha_0 = -1$, e $\alpha_{-1} = 0$. Durran e Blossey (2012) mostram que essa classe particular de métodos possui propriedades de estabilidade que a torna atraente para a aproximação IMEX da solução de problemas relacionados à dinâmica da atmosfera em que existem ondas lentas e rápidas. Deve-se notar que método explícito de ordem mais alta que pode ser obtido utilizando s níveis é o método de Adams-Bashforth de s-nível, enquanto o método implícito de ordem mais alta é o método de Adams-Moulton de s-nível. No entanto, o único método de Adams-Moulton que é A-estável é incapaz de amortecer oscilações de alta frequência, e portanto os autores preferem considerar métodos com uma ordem de acurácia menor do que o máximo possível.

A estratégia de integração explícita baseada nos métodos de Adams é descrita por uma equação com um parâmetro numérico, b, dada por:

$$f(u) \approx \frac{(3+b)}{2} f(u^n) - \frac{(1+2b)}{2} f(u^{n-1}) + \frac{b}{2} f(u^{n-2}) \tag{2.22}$$

De forma equivalente, a estratégia implícita baseada nos métodos de Adams é dada

por outra equação com um parâmetro numérico, c, dada por:

$$g(u) \approx \frac{(1+c)}{2} g(u^{n+1}) + \frac{(1-2c)}{2} g(u^n) + \frac{c}{2} g(u^{n-1}) \tag{2.23}$$

Três combinações de métodos desse tipo são propostas pelos autores. Seus parâmetros correspondentes, bem como a nomenclatura associada a cada método, são exibidos na Tabela 2.1.

Tabela 2.1 - Parâmetros para esquemas IMEX *multistep* baseados em métodos de Adams.

Implícito	Explícito	b	c	β_0	β_{-1}	β_{-2}	λ_1	λ_0	λ_{-1}
MCN	AX2+	3/8	1/8	27/16	-7/8	3/16	9/16	3/8	1/16
AM2*	AX2*	1/2	1/2	7/4	-1	1/4	3/4	0	1/4
AI2*	AB3	5/6	3/2	23/12	-4/3	5/12	5/4	-1	3/4

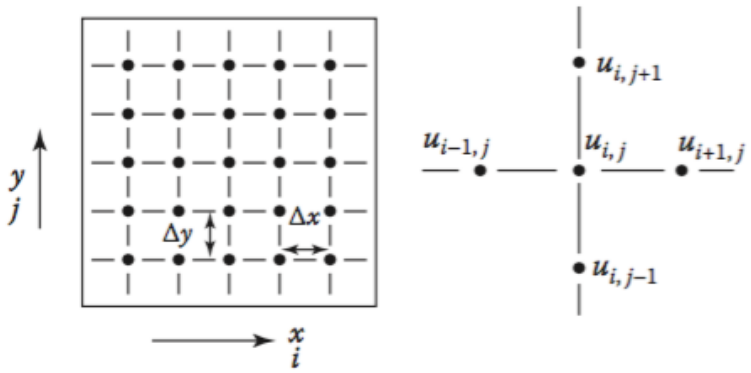
Fonte: Adaptado de Durran e Blossey (2012).

2.3 Discretizações espaciais

Discretizações para as derivadas espaciais em uma EDP são realizadas seguindo a mesma metodologia apresenta na seção 2.1, independentemente do número de coordenadas espaciais. Considere por exemplo uma função $u(x, y)$ em domínio contínuo. A grade é estabelecida por meio da sua substituição por $u(i\Delta x, j\Delta y)$, onde i e j são números inteiros utilizados para localizar os pontos desejados no espaço e Δx e Δy são os incrementos em suas respectivas direções (x, y) entre cada ponto de grade. A Figura 2.1 ilustra uma grade típica de um método de diferenças finitas, bem como a nomenclatura utilizada para localização dos pontos – em que a notação $u_{i,j}$ representa o ponto $u(i\Delta x, j\Delta y)$.

Seguindo a metodologia proposta na seção 2.1, podemos então obter as discretizações progressiva, regressiva e central, respectivamente, para as derivadas parciais de

Figura 2.1 - Grade típica do método de diferenças finitas.



Fonte: Anderson et al. (2012), p. 44.

primeira ordem em x:

$$\begin{aligned}\left(\frac{\partial u}{\partial x}\right)_{i,j} &= \frac{u_{i+1,j}-u_{i,j}}{\Delta x}+O(\Delta x)\\ \left(\frac{\partial u}{\partial x}\right)_{i,j} &= \frac{u_{i,j}-u_{i-1,j}}{\Delta x}+O(\Delta x)\\ \left(\frac{\partial u}{\partial x}\right)_{i,j} &= \frac{u_{i+1,j}-u_{i-1,j}}{2\Delta x}+O(\Delta x)^2\end{aligned} \tag{2.24}$$

e, de forma análoga, em y:

$$\begin{aligned}\left(\frac{\partial u}{\partial y}\right)_{i,j} &= \frac{u_{i,j+1}-u_{i,j}}{\Delta y}+O(\Delta y)\\ \left(\frac{\partial u}{\partial y}\right)_{i,j} &= \frac{u_{i,j}-u_{i,j-1}}{\Delta y}+O(\Delta y)\\ \left(\frac{\partial u}{\partial y}\right)_{i,j} &= \frac{u_{i,j+1}-u_{i,j-1}}{2\Delta y}+O(\Delta y)^2\end{aligned} \tag{2.25}$$

Para as derivadas de segunda ordem, considere a série de Taylor representada pela equação 2.2, aqui repetida:

$$u_{i+1}=u_i+\Delta x\left(\frac{du}{dx}\right)_i+\frac{(\Delta x)^2}{2!}\left(\frac{d^2u}{dx^2}\right)_i+\frac{(\Delta x)^3}{3!}\left(\frac{d^3u}{dx^3}\right)_i+\ldots$$

e também a diferença finita central de segunda ordem, fornecida pela equação 2.8:

$$\left(\frac{du}{dx}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x)^2$$

Substituindo a primeira derivada em 2.2 pela aproximação 2.8:, obtém-se:

$$\begin{aligned} u_{i+1} = u_i + \Delta x \left[\frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x)^2\right] + \\ \frac{(\Delta x)^2}{2!}\left(\frac{d^2u}{dx^2}\right)_i + \frac{(\Delta x)^3}{3!}\left(\frac{d^3u}{dx^3}\right)_i + ... \end{aligned} \tag{2.26}$$

Resolvendo para a segunda derivada, obtemos a diferença finita central de segunda ordem:

$$\left(\frac{d^2u}{dx^2}\right)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} + O(\Delta x)^2 \tag{2.27}$$

Escrevemos então as derivadas parciais de segunda ordem para x:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + O(\Delta x)^2 \tag{2.28}$$

e y:

$$\left(\frac{\partial^2 u}{\partial y^2}\right)_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} + O(\Delta y)^2 \tag{2.29}$$

Novamente, deve-se mencionar que aproximações de ordem mais alta (e também derivadas de ordens mais altas) podem ser obtidas utilizando o mesmo princípio. Seria inviável demonstrar neste trabalho todas as possibilidades, porém um grande número de discretizações possíveis pode ser encontrado em Anderson et al. (2012, pp. 48-50).

2.4 Considerações sobre estabilidade numérica

A estabilidade de esquemas numéricos é relacionada ao erro numérico produzido em um passo de tempo, que causa um desvio em relação à solução esperada. Podemos definir o fator de amplificação, A, como a razão entre as soluções aproximadas em dois passos de tempo adjacentes (DURRAN, 2010):

$$A = \frac{\phi_{n+1}}{\phi_n} \tag{2.30}$$

Durran (2010) define que um método de dois níveis temporais é estável no sentido que, caso também seja consistente, ele converge no limite em que $\Delta t \to 0$ desde que para uma constante η (independente das propriedades da discretização numérica):

$$|A| \leqslant 1 + \eta \Delta t \tag{2.31}$$

É imediatamente notável que, nos casos em que $|A| > 1$, a solução aumenta (ou diminui) a cada passo, tendendo ao infinito após um número suficiente de passos temporais. De forma equivalente, mesmo em casos em que $|A| \leqslant 1$, é possível que a solução seja inflada de forma contrária à física do problema quando computada com um valor finito de Δt.

O conceito de A-estabilidade é introduzido para caracterizar a estabilidade de um método numérico consistente, garantindo que as soluções não irão inflar de forma indesejável em problemas práticos. Durran (2010) considera então um simples caso teste,

$$\frac{d\psi}{dt} = \gamma\psi, \quad \psi(0) = \psi_0 \tag{2.32}$$

onde ψ e γ possuem valores complexos. Dividindo γ em suas partes real e imaginária, de forma que $\psi = \lambda + i\omega$ (com λ e ω reais), a solução da Equação 2.32 é dada por:

$$\psi(t) = \psi_0 e^{\lambda t} e^{i\omega t} \tag{2.33}$$

Soluções numéricas do problema descrito pela Equação 2.32 computadas com um valor finito de Δt são consideradas absolutamente estáveis se $|\phi_n| \leqslant |\phi_0|$, ou, de forma equivalente, se $|A| \leqslant 1$.

A solução real da Equação 2.32 não é amplificada para nenhum $\lambda > 0$. Durran (2010) postula então que esse comportamente será capturado independentemente do passo de tempo por métodos numéricos A-estáveis. Um método é considerado A-estável quando ele é absolutamente estável para todo $\lambda\Delta t \leqslant 0$.

Embora estes conceitos propostos por Durran (2010) sejam aplicados a esquemas de dois níveis, Eilbeck e McGuire (1975) notam que esquemas com três níveis temporais podem ser expressados como um sistema de esquemas de dois níveis, possibilitando a aplicação deste tipo de análise.

Já o cálculo do fator de amplificação é frequentemente realizado por meio da análise de von Neumann, conforme descrito por Richtmyer e Morton (1967). A princípio, a análise de von Neumann é aplicável apenas a equações lineares em problemas de valor

inicial ou periódicos. No entanto, é possível aplicá-la a certos problemas não lineares desde que seja feita uma linearização dos termos não lineares (EILBECK; MCGUIRE, 1975), considerando um valor característico para as variáveis linearizadas. A partir das novas equações, é possível calcular um fator de amplificação para análise do problema. Esse nível de detalhamento analítico está fora do escopo desta dissertação, e portanto não será descrito aqui.

3 EQUAÇÃO DE BURGERS

A equação de Burgers é uma das equações fundamentais da dinâmica de fluidos computacional. Trata-se de uma equação diferencial parcial parabólica, capaz de modelar efetivamente certos problemas relacionados ao fluxo de fluidos influenciados por ondas de choque ou dissipação viscosa. Ela serve como um modelo para equações de camada limite (HUGHSON et al., 1987) e também como uma forma especial das equações de Navier-Stokes incompressíveis, ignorando o termo de pressão e a equação de continuidade (SRIVASTAVA et al., 2011)

Introduzida em sua forma unidimensional por Harry Bateman (BATEMAN, 1915) para descrever o fluxo de certos fluidos viscosos, a equação ganhou seu nome anos mais tarde, em 1948, quando Johannes Martinus Burgers a utilizou como base para sua modelagem matemática de turbulência (BURGERS, 1948). Em sua forma mais simples, a equação de Burgers é dada por:

$$u_t + uu_x = \nu u_{xx} \tag{3.1}$$

em que ν é o coeficiente de difusão (ou, no contexto da dinâmica de fluidos, o coeficiente de viscosidade). Desde então, ela foi empregada na modelagem de diversos fenômenos físicos, como fluxo de tráfego, formação de ondas de choque, transporte de massa, e outros. Seu uso na modelagem do fluxo de fluidos viscosos baseia-se do fato de que ela incorpora diretamente a interação entre os processos convectivos não lineares e os processos difusivos viscosos (FLETCHER, 1983).

É possível separar os termos da Equação 3.1 em seus componentes lentos (convectivos) e rápidos (difusivos), representados aqui por $f(u)$ e $g(u)$ respectivamente. A equação de Burgers toma então a forma:

$$u_t = f(u) + \nu g(u) \tag{3.2}$$

onde

$$f(u) = -uu_x \tag{3.3}$$

$$g(u) = u_{xx} \tag{3.4}$$

A Equação 3.2 é equivalente à forma canônica dos esquemas IMEX, conforme a Equação 2.19, e portanto uma boa candidata a esse tipo de integração temporal.

Embora não tenha sido resolvida de forma fechada, é possível encontrar soluções

exatas para muitas combinações de condições iniciais e de contorno (SMITH III, 1997) através da transformada de Hopf-Cole. Kweyu et al. (2012) demonstram o processo de obtenção dessas soluções, reproduzido com anotações no Apêndice A.

Por todos esses motivos, e também devido à sua simplicidade, a equação de Burgers encontrou uso extensivo no teste de novos algoritmos numéricos (LIU; WANG, 2015) e na comparação entre diferentes algoritmos. Sendo assim, este trabalho a adota como o modelo teste para a família de esquemas IMEX propostos no Capítulo 2.

Em duas dimensões, a equação 3.1 se transforma em um sistema de duas equações acopladas que representam uma forma especial das equações de Navier-Stokes incompressíveis, ignorando o termo de pressão e a equação de continuidade (SRIVASTAVA et al., 2011). O sistema é dado por:

$$u_t + uu_x + vu_y = \nu(u_{xx} + u_{yy}) \tag{3.5}$$

$$v_t + uv_x + vv_y = \nu(v_{xx} + v_{yy}) \tag{3.6}$$

e está sujeito às condições iniciais

$$\left.\begin{aligned} u(x,y,0) &= \varphi_1(x,y) \\ v(x,y,0) &= \varphi_2(x,y) \end{aligned}\right\} (x,y) \in \Omega \tag{3.7}$$

e às condições de contorno

$$\left.\begin{aligned} u(x,y,t) &= \xi_1(x,y,t) \\ v(x,y,t) &= \xi_2(x,y,t) \end{aligned}\right\} (x,y) \in \partial\Omega, \quad t > 0 \tag{3.8}$$

onde ν é o coeficiente de viscosidade, dado pelo inverso do número de Reynolds (R_e); $\Omega = \{(x,y) : a \leqslant x \leqslant b, \quad c \leqslant y \leqslant d\}$ representa o domínio dos valores assumidos por x e y; e $d\Omega$ representa suas bordas.

De forma equivalente ao procedimento realizado para a equação unidimensional, as Equações 3.5 e 3.6 podem ser reescritas na forma:

$$u_t = -(uu_x + vu_y) + \nu(u_{xx} + u_{yy}) \tag{3.9}$$

$$v_t = -(uv_x + vv_y) + \nu(v_{xx} + v_{yy}) \tag{3.10}$$

Seja Φ o vetor de velocidades, dado em duas dimensões por:

$$\Phi = \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.11}$$

É possível então escrever a equação de Burgers bidimensional na forma matricial,

$$\Phi_t = f(\Phi) + \nu g(\Phi) \tag{3.12}$$

onde

$$f(\Phi) = -\frac{1}{2}\nabla(\Phi \cdot \Phi) \tag{3.13}$$

$$g(\Phi) = \nabla^2 \Phi \tag{3.14}$$

Novamente, o sistema obtido é compartível com a forma canônica dos esquemas IMEX, conforme descrito por Ascher et al. (1995). Deste ponto em diante, o texto tratará a Equação 3.12 como representativa dos dois casos (unidimensional e bidimensional), já que ela é equivalente à Equação 3.2 quando o vetor Φ possui apenas uma componente de velocidade (u).

3.1 Discretizações espaciais para equação de Burgers

As discretizações espaciais utilizadas neste trabalho seguem o modelo de diferenças centrais apresentado no Capítulo 2. Para o caso unidimensional, são utilizadas:

$$u_x \approx \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} \tag{3.15}$$

$$u_{xx} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{3.16}$$

onde u_i^n denota $u(x_i, t_n)$.

De forma equivalente, o caso bidimensional utiliza:

$$u_x \approx \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2\Delta x} \tag{3.17}$$

$$u_y \approx \frac{u_{i,j+1}^n - u_{i,j-1}^n}{2\Delta y} \tag{3.18}$$

$$u_{xx} \approx \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} \tag{3.19}$$

$$u_{yy} \approx \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta y^2} \tag{3.20}$$

onde $u^n_{i,j}$ denota $u(x_i, y_j, t_n)$.

3.2 Abordagem numérica explícita com esquema FTCS

A primeira implementação da solução numérica do sistema de Burgers utiliza um esquema explícito para discretizar as equações. Especificamente, foi escolhido o esquema FTCS (*forward-time, central-space*) no qual a discretização temporal é feita por meio de diferenças finitas progressivas e a discretização espacial é feita por diferenças finitas centradas.

$$\frac{\Phi^{n+1} - \Phi^n}{\Delta t} = f(\Phi^n) + \nu g(\Phi^n) \tag{3.21}$$

No caso unidimensional, a equação discretizada completa é dada por:

$$u^{n+1}_i = u^n_i - \frac{\Delta t}{2\Delta x} u^n_i (u^n_{i+1} - u^n_{i-1}) + \frac{\nu \Delta t}{\Delta x^2}(u^n_{i+1} - 2u^n_i + u^n_{i-1}) \tag{3.22}$$

No caso bidimensional, a discretização completa da equação 3.5 é dada por:

$$\begin{aligned} \frac{u^{n+1}_{i,j} - u^n_{i,j}}{\Delta t} = &-u^n_{i,j}\left(\frac{u^n_{i+1,j} - u^n_{i-1,j}}{2\Delta x}\right) - v^n_{i,j}\left(\frac{u^n_{i,j+1} - u^n_{i,j-1}}{2\Delta y}\right) \\ &+\nu\left[\left(\frac{u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}}{\Delta x^2}\right) + \left(\frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta y^2}\right)\right] \end{aligned} \tag{3.23}$$

e, de forma similar para a equação 3.6,

$$\begin{aligned} \frac{v^{n+1}_{i,j} - v^n_{i,j}}{\Delta t} = &-u^n_{i,j}\left(\frac{v^n_{i+1,j} - v^n_{i-1,j}}{2\Delta x}\right) - v^n_{i,j}\left(\frac{v^n_{i,j+1} - v^n_{i,j-1}}{2\Delta y}\right) + \\ &\nu\left[\left(\frac{v^n_{i+1,j} - 2v^n_{i,j} + v^n_{i-1,j}}{\Delta x^2}\right) + \left(\frac{v^n_{i,j+1} - 2v^n_{i,j} + v^n_{i,j-1}}{\Delta y^2}\right)\right] \end{aligned} \tag{3.24}$$

Em cada passo de tempo são calculados os valores para $u^{n+1}_{i,j}$ e $v^{n+1}_{i,j}$ para todos os pares (i, j) não pertencentes à borda, utilizando como base os valores calculados no passo anterior. Os elementos da borda são fixados com base nas condições de contorno.

A solução da equação de Burgers pelo método FTCS é estudada no trabalho de Kweyu et al. (2012), permitindo a validação da implementação proposta através de casos de estudo no Capítulo 4.

3.3 Abordagem numérica implícita com método de Crank-Nicolson

A segunda implementação numérica utiliza o método de Crank-Nicolson para criar uma discretização implícita. O método, inicialmente proposto por Crank e Nicolson (1947), consiste na utilização a regra trapezoidal (conforme definida na equação 2.13) para a discretização temporal. Seu uso na solução da equação de Burgers foi estudado em Srivastava et al. (2011).

O esquema geral de integração é dado por:

$$\frac{\Phi^{n+1}-\Phi^{n}}{\Delta t}=\frac{1}{2}[f(\Phi^{n+1})+f(\Phi^{n})]+\frac{\nu}{2}[g(\Phi^{n+1})+g(\Phi^{n})]\ . \tag{3.25}$$

Aplicando diferenças centrais de segunda ordem aos termos espaciais, chega-se à discretização completa via Crank-Nicolson para a forma unidimensional:

$$\begin{aligned} u_i^{n+1} = u_i^n &- \frac{\Delta t}{4\Delta x}u_i^{n+1}(u_{i+1}^{n+1}-u_{i-1}^{n+1}) - \frac{\Delta t}{4\Delta x}u_i^n(u_{i+1}^n-u_{i-1}^n) \\ &+\frac{\nu\Delta t}{2\Delta x^2}(u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}) + \frac{\nu\Delta t}{2\Delta x^2}(u_{i+1}^n-2u_i^n+u_{i-1}^n) \end{aligned} \tag{3.26}$$

Analogamente, no caso bidimensional, é feita a discretização da equação 3.5:

$$\begin{aligned} \frac{u_{i,j}^{n+1}-u_{i,j}^n}{\Delta t} = &-\frac{1}{2}\left[u_{i,j}^{n+1}\left(\frac{u_{i+1,j}^{n+1}-u_{i-1,j}^{n+1}}{2\Delta x}\right)+u_{i,j}^n\left(\frac{u_{i+1,j}^n-u_{i-1,j}^n}{2\Delta x}\right)\right] \\ &-\frac{1}{2}\left[v_{i,j}^{n+1}\left(\frac{u_{i,j+1}^{n+1}-u_{i,j-1}^{n+1}}{2\Delta y}\right)+v_{i,j}^n\left(\frac{u_{i,j+1}^n-u_{i,j-1}^n}{2\Delta y}\right)\right] \\ &+\frac{\nu}{2}\left[\left(\frac{u_{i+1,j}^{n+1}-2u_{i,j}^{n+1}+u_{i-1,j}^{n+1}}{\Delta x^2}\right)+\left(\frac{u_{i+1,j}^n-2u_{i,j}^n+u_{i-1,j}^n}{\Delta x^2}\right)\right] \\ &+\frac{\nu}{2}\left[\left(\frac{u_{i,j+1}^{n+1}-2u_{i,j}^{n+1}+u_{i,j-1}^{n+1}}{\Delta y^2}+\left(\frac{u_{i,j+1}^n-2u_{i,j}^n+u_{i,j-1}^n}{\Delta y^2}\right)\right)\right] \end{aligned} \tag{3.27}$$

e da equação 3.6:

$$\begin{aligned} \frac{v_{i,j}^{n+1}-v_{i,j}^n}{\Delta t} = &-\frac{1}{2}\left[u_{i,j}^{n+1}\left(\frac{v_{i+1,j}^{n+1}-v_{i-1,j}^{n+1}}{2\Delta x}\right)+u_{i,j}^n\left(\frac{v_{i+1,j}^n-v_{i-1,j}^n}{2\Delta x}\right)\right] \\ &-\frac{1}{2}\left[v_{i,j}^{n+1}\left(\frac{v_{i,j+1}^{n+1}-v_{i,j-1}^{n+1}}{2\Delta y}\right)+v_{i,j}^n\left(\frac{v_{i,j+1}^n-v_{i,j-1}^n}{2\Delta y}\right)\right] \\ &+\frac{\nu}{2}\left[\left(\frac{v_{i+1,j}^{n+1}-2v_{i,j}^{n+1}+v_{i-1,j}^{n+1}}{\Delta x^2}\right)+\left(\frac{v_{i+1,j}^n-2v_{i,j}^n+v_{i-1,j}^n}{\Delta x^2}\right)\right] \\ &+\frac{\nu}{2}\left[\left(\frac{v_{i,j+1}^{n+1}-2v_{i,j}^{n+1}+v_{i,j-1}^{n+1}}{\Delta y^2}+\left(\frac{v_{i,j+1}^n-2v_{i,j}^n+v_{i,j-1}^n}{\Delta y^2}\right)\right)\right] \end{aligned} \tag{3.28}$$

No caso unidimensional, a discretização implícita gera um sistema não linear com n_x equações que devem ser resolvidas a cada passo de tempo, onde n_x é o número de pontos ao longo do eixo x. Cada equação do sistema toma a forma:

$$\begin{aligned}\left[1+\frac{\nu\Delta t}{\Delta x^2}\right]u_i^{n+1}+\left[\frac{\Delta t}{4\Delta x}\right]u_i^{n+1}(u_{i+1}^{n+1}-u_{i-1}^{n+1})-\left[\frac{\nu\Delta t}{2\Delta x^2}\right](u_{i+1}^{n+1}+u_{i-1}^{n+1})\\ =u_i^n-\frac{\Delta t}{4\Delta x}u_i^n(u_{i+1}^n-u_{i-1}^n)+\frac{\nu\Delta t}{2\Delta x^2}(u_{i+1}^n-2u_i^n+u_{i-1}^n)\end{aligned} \tag{3.29}$$

Em duas dimensões, o sistema contém $2n_xn_y$ equações, sendo n_y o número de pontos ao longo do eixo y, pois o acoplamento do sistema requer que as duas componentes sejam resolvidas simultaneamente. Considerando-se as equações 3.27 e 3.28 como um sistema de equações não lineares na forma:

$$\Phi(w)=0, \tag{3.30}$$

em que:

$$\begin{aligned}\Phi=[(\phi_u)_{1,1},(\phi_v)_{1,1},...,(\phi_u)_{1,ny},(\phi_v)_{1,ny},...,(\phi_u)_{nx,ny},(\phi_v)_{nx,ny}]^T\\ w=[u_{1,1}^{n+1},v_{1,1}^{n+1},...,u_{1,ny}^{n+1},v_{1,ny}^{n+1},...,u_{nx,ny}^{n+1},v_{nx,ny}^{n+1}]^T\end{aligned} \tag{3.31}$$

Os elementos $(\phi_u)_{i,j}$ e $(\phi_v)_{i,j}$ de Φ representam as discretizações de $u_{i,j}^{n+1}$ e $v_{i,j}^{n+1}$, respectivamente. A solução desse sistema não linear pode ser aproximada de forma iterativa pelo método de Newton. Aplicado ao sistema 3.30, as iterações consistem em:

a) determinar uma aproximação inicial, w^0,

b) para $k=0,1,...$ até atingir a convergência:

- resolver o sistema linear $J(w^{(k)})\times\delta^{(k)}=-\Phi(w^{(k)})$
- gerar uma nova aproximação, $w^{(k+1)}=w^{(k)}+\delta^{(k)}$.

No esquema descrito acima, $J(w^{(k)})$ é a matriz jacobiana e $\delta^{(k)}$ é o vetor de correção para aproximação $w^{(k)}$. Os termos da matriz jacobiana podem ser encontrados analiticamente derivando cada elemento de Φ por cada elemento de w, tomando a seguinte forma:

$$J(w) = \begin{bmatrix} \frac{\partial(\phi_u)_{1,1}}{\partial u_{1,1}} & \frac{\partial(\phi_u)_{1,1}}{\partial v_{1,1}} & \cdots & \frac{\partial(\phi_u)_{1,1}}{\partial u_{nx,ny}} & \frac{\partial(\phi_u)_{1,1}}{\partial v_{nx,ny}} \\ \frac{\partial(\phi_v)_{1,1}}{\partial u_{1,1}} & \frac{\partial(\phi_v)_{1,1}}{\partial v_{1,1}} & \cdots & \frac{\partial(\phi_v)_{1,1}}{\partial u_{nx,ny}} & \frac{\partial(\phi_v)_{1,1}}{\partial v_{nx,ny}} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \frac{\partial(\phi_u)_{nx,ny}}{\partial u_{1,1}} & \frac{\partial(\phi_u)_{nx,ny}}{\partial v_{1,1}} & \cdots & \frac{\partial(\phi_u)_{nx,ny}}{\partial u_{nx,ny}} & \frac{\partial(\phi_u)_{nx,ny}}{\partial v_{nx,ny}} \\ \frac{\partial(\phi_v)_{nx,ny}}{\partial u_{1,1}} & \frac{\partial(\phi_v)_{nx,ny}}{\partial v_{1,1}} & \cdots & \frac{\partial(\phi_v)_{nx,ny}}{\partial u_{nx,ny}} & \frac{\partial(\phi_v)_{nx,ny}}{\partial v_{nx,ny}} \end{bmatrix} \tag{3.32}$$

A Tabela 3.1 apresenta, em termos genéricos, as derivadas de ϕ_u e ϕ_v com relação a cada elemento relacionado de w.

Tabela 3.1 - Termos da matriz jacobiana para esquema de Crank-Nicolson.

	$\partial(\phi_u)_{i,j}$	$\partial(\phi_v)_{i,j}$
$\partial u_{i,j}$	$\frac{1}{\Delta t} + \frac{\nu}{\Delta x^2} + \frac{\nu}{\Delta y^2} + \frac{(u_{i+1,j}^{n+1} - u_{i-1,j}^{n+1})}{4\Delta x}$	$\frac{(v_{i+1,j}^{n+1} - v_{i-1,j}^{n+1})}{4\Delta x}$
$\partial v_{i,j}$	$\frac{(u_{i,j+1}^{n+1} - u_{i,j-1}^{n+1})}{4\Delta y}$	$\frac{1}{\Delta t} + \frac{\nu}{\Delta x^2} + \frac{\nu}{\Delta y^2} + \frac{(v_{i,j+1}^{n+1} - v_{i,j-1}^{n+1})}{4\Delta y}$
$\partial u_{i+1,j}$	$\frac{u_{i,j}^{n+1}}{4\Delta x} - \frac{\nu}{2\Delta x^2}$	0
$\partial v_{i+1,j}$	0	$\frac{u_{i,j}^{n+1}}{4\Delta x} - \frac{\nu}{2\Delta x^2}$
$\partial u_{i-1,j}$	$-\frac{u_{i,j}^{n+1}}{4\Delta x} - \frac{\nu}{2\Delta x^2}$	0
$\partial v_{i-1,j}$	0	$-\frac{u_{i,j}^{n+1}}{4\Delta x} - \frac{\nu}{2\Delta x^2}$
$\partial u_{i,j+1}$	$\frac{v_{i,j}^{n+1}}{4\Delta y} - \frac{\nu}{2\Delta y^2}$	0
$\partial v_{i,j+1}$	0	$\frac{v_{i,j}^{n+1}}{4\Delta y} - \frac{\nu}{2\Delta y^2}$
$\partial u_{i,j-1}$	$-\frac{v_{i,j}^{n+1}}{4\Delta y} - \frac{\nu}{2\Delta y^2}$	0
$\partial v_{i,j-1}$	0	$-\frac{v_{i,j}^{n+1}}{4\Delta y} - \frac{\nu}{2\Delta y^2}$

Em cada iteração os elementos da matriz J e os do vetor Φ são recalculados com base nos valores da aproximação atual, w^k, e o sistema linear é resolvido via fatorização LU gerando um novo vetor de correção δ. Deve-se notar que a escolha da aproximação inicial para o método de Newton é extremamente importante, pois a convergência pode não ocorrer se ela estiver muito distante da solução (BAHADIR, 2003). Para garantir uma convergência rápida, a aproximação inicial escolhida é a solução numérica do passo de tempo anterior.

3.4 Abordagem numérica com esquemas IMEX *multistep*

A abordagem IMEX *multistep* proposta no capítulo anterior envolve a aplicação da equação 2.22 ao termo explícito, $f(u)$, e da equação 2.23 ao termo implícito, $g(u)$.

A estratégia geral de integração da equação de Burgers por este método é dada por:

$$\begin{aligned}\frac{\Phi^{n+1}-\Phi^{n}}{\Delta t} &= \frac{(3+b)}{2}f(\Phi^{n}) - \frac{(1+2b)}{2}f(\Phi^{n-1}) + \frac{b}{2}f(\Phi^{n-2}) \\ &+\nu\left[\frac{(1+c)}{2}g(\Phi^{n+1}) + \frac{(1-2c)}{2}g(\Phi^{n}) + \frac{c}{2}g(\Phi^{n-1})\right]\end{aligned} \tag{3.33}$$

onde a escolha dos parâmetros b e c define o esquema resultante. A Tabela 2.1 contém três possíveis combinações de métodos de Adams, conforme o trabalho de Durran e Blossey (2012).

No primeiro caso, a escolha de $b = \frac{3}{8}$ e $c = \frac{1}{8}$ gera o método denominado MCN-AX2+ para a solução da equação de Burgers:

$$\begin{aligned}\frac{\Phi^{n+1}-\Phi^{n}}{\Delta t} &= \frac{27}{16}f(\Phi^{n}) + \frac{27}{8}f(\Phi^{n-1}) + \frac{3}{16}f(\Phi^{n-2}) \\ &+\nu\left[\frac{9}{16}g(\Phi^{n+1}) + \frac{3}{8}g(\Phi^{n}) + \frac{1}{16}g(\Phi^{n-1})\right]\end{aligned} \tag{3.34}$$

Escolhendo $b = \frac{1}{2}$ and $c = \frac{1}{2}$, obtém-se o método AM2*-AX2*+:

$$\begin{aligned}\frac{\Phi^{n+1}-\Phi^{n}}{\Delta t} &= \frac{7}{4}f(\Phi^{n}) - f(\Phi^{n-1}) + \frac{1}{4}f(\Phi^{n-2}) \\ &+\nu\left[\frac{3}{4}g(\Phi^{n+1}) + \frac{1}{4}g(\Phi^{n-1})\right]\end{aligned} \tag{3.35}$$

O método denominado AI2*-AB3 é obtido quando $b = \frac{5}{6}$ e $c = \frac{3}{2}$. Sua forma geral é dada por:

$$\begin{aligned}\frac{\Phi^{n+1}-\Phi^{n}}{\Delta t} &= \frac{23}{12}f(\Phi^{n}) - \frac{4}{3}f(\Phi^{n-1}) + \frac{5}{12}f(\Phi^{n-2}) \\ &+\nu\left[\frac{5}{4}g(\Phi^{n+1}) - g(\Phi^{n}) + \frac{3}{4}g(\Phi^{n-1})\right]\end{aligned} \tag{3.36}$$

Para as discretizações espaciais, propõe-se a utilização de diferenças finitas centrais de segunda ordem, mantendo assim a mesma metodologia utilizada nos esquemas explícito e implícito já implementados.

Considerando que a única diferença entre os três esquemas é a escolha dos parâmetros numéricos b e c, é possível criar uma implementação generalizada que leve em conta

esses fatores. Assim, o caso unidimensional pode ser representado por:

$$\begin{aligned} u_i^{n+1} &= u_i^n \\ &- \left[\frac{(3+b)}{4}\frac{\Delta t}{\Delta x}\right] u_i^n (u_{i+1}^n - u_{i-1}^n) + \left[\frac{(1+2b)}{4}\frac{\Delta t}{\Delta x}\right] u_i^{n-1}(u_{i+1}^{n-1} - u_{i-1}^{n-1}) \\ &- \left[\frac{b}{4}\frac{\Delta t}{\Delta x}\right] u_i^{n-2}(u_{i+1}^{n-2} - u_{i-1}^{n-2}) + \left[\frac{(1+c)}{2}\frac{\nu\Delta t}{\Delta x^2}\right](u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) \\ &+ \left[\frac{(1-2c)}{2}\frac{\nu\Delta t}{\Delta x^2}\right](u_{i+1}^{n} - 2u_i^{n} + u_{i-1}^{n}) + \left[\frac{c}{2}\frac{\nu\Delta t}{\Delta x^2}\right](u_{i+1}^{n-1} - 2u_i^{n-1} + u_{i-1}^{n-1}) \end{aligned} \tag{3.37}$$

Analogamente, discretizações para u e v no caso bidimensional podem ser representadas pelas equações 3.38 e 3.39, respectivamente.

$$\begin{aligned} u_{i,j}^{n+1} &= u_{i,j}^n \\ &- \frac{(3+b)}{4}\Delta t\left[\frac{u_{i,j}^n(u_{i+1,j}^n - u_{i-1,j}^n)}{\Delta x} + \frac{v_{i,j}^n(u_{i,j+1}^n - u_{i,j-1}^n)}{\Delta y}\right] \\ &+ \frac{(1+2b)}{4}\Delta t\left[\frac{u_{i,j}^{n-1}(u_{i+1,j}^{n-1} - u_{i-1,j}^{n-1})}{\Delta x} + \frac{v_{i,j}^{n-1}(u_{i,j+1}^{n-1} - u_{i,j-1}^{n-1})}{\Delta y}\right] \\ &- \frac{b}{4}\Delta t\left[\frac{u_{i,j}^{n-2}(u_{i+1,j}^{n-2} - u_{i-1,j}^{n-2})}{\Delta x} + \frac{v_{i,j}^{n-2}(u_{i,j+1}^{n-2} - u_{i,j-1}^{n-2})}{\Delta y}\right] \\ &+ \frac{(1+c)}{2}\nu\Delta t\left[\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2}\right] \\ &+ \frac{(1-2c)}{2}\nu\Delta t\left[\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2}\right] \\ &+ \frac{c}{2}\nu\Delta t\left[\frac{u_{i+1,j}^{n-1} - 2u_{i,j}^{n-1} + u_{i-1,j}^{n-1}}{\Delta x^2} + \frac{u_{i,j+1}^{n-1} - 2u_{i,j}^{n-1} + u_{i,j-1}^{n-1}}{\Delta y^2}\right] \end{aligned} \tag{3.38}$$

$$\begin{aligned} v_{i,j}^{n+1} &= v_{i,j}^n \\ &- \frac{(3+b)}{4}\Delta t\left[\frac{u_{i,j}^n(v_{i+1,j}^n - v_{i-1,j}^n)}{\Delta x} + \frac{v_{i,j}^n(v_{i,j+1}^n - v_{i,j-1}^n)}{\Delta y}\right] \\ &+ \frac{(1+2b)}{4}\Delta t\left[\frac{u_{i,j}^{n-1}(v_{i+1,j}^{n-1} - v_{i-1,j}^{n-1})}{\Delta x} + \frac{v_{i,j}^{n-1}(v_{i,j+1}^{n-1} - v_{i,j-1}^{n-1})}{\Delta y}\right] \\ &- \frac{b}{4}\Delta t\left[\frac{u_{i,j}^{n-2}(v_{i+1,j}^{n-2} - v_{i-1,j}^{n-2})}{\Delta x} + \frac{v_{i,j}^{n-2}(v_{i,j+1}^{n-2} - v_{i,j-1}^{n-2})}{\Delta y}\right] \\ &+ \frac{(1+c)}{2}\nu\Delta t\left[\frac{v_{i+1,j}^{n+1} - 2v_{i,j}^{n+1} + v_{i-1,j}^{n+1}}{\Delta x^2} + \frac{v_{i,j+1}^{n+1} - 2v_{i,j}^{n+1} + v_{i,j-1}^{n+1}}{\Delta y^2}\right] \\ &+ \frac{(1-2c)}{2}\nu\Delta t\left[\frac{v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}}{\Delta x^2} + \frac{v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n}}{\Delta y^2}\right] \\ &+ \frac{c}{2}\nu\Delta t\left[\frac{v_{i+1,j}^{n-1} - 2v_{i,j}^{n-1} + v_{i-1,j}^{n-1}}{\Delta x^2} + \frac{v_{i,j+1}^{n-1} - 2v_{i,j}^{n-1} + v_{i,j-1}^{n-1}}{\Delta y^2}\right] \end{aligned} \tag{3.39}$$

É importante notar que os esquemas propostos necessitam de informações de dois passos de tempo anteriores (n^{-1}, n^{-2}); sendo assim, é necessário gerar essas informações por meio de algum outro método. A implementação realizada no desenvolvimento desta dissertação utiliza o método FTCS para a solução dos dois passos iniciais ($n = 2$ e $n = 3$, sendo $n = 1$ a condição inicial). Em casos nos quais o método FTCS é instável (seja por causa de alta resolução espacial ou de um passo de tempo largo), deve-se utilizar passos de tempo pequenos o suficiente para garantir sua estabilidade durante esse período de *warm-up*.

3.4.1 Inicialização e solução de sistemas lineares na abordagem IMEX

Dado o tratamento explícito dos termos não lineares, o sistema de equações resultante em casa passo de tempo desta abordagem é linear, e portanto não necessita de um método iterativo de aproximação. Em duas dimensões, a linearização do sistema tem o efeito adicional de desacoplar as soluções de u e v ao nível do passo de tempo, como observado nas equações 3.38 (que contém apenas a variável u no passo de tempo $n + 1$) e 3.39 (que contém apenas v).

Os sistemas gerados possuem uma propriedade notável: a matriz de coeficientes é a mesma tanto para u e v, e seus elementos são constantes numéricas independentes dos campos de velocidade. No caso unidimensional, essencialmente tem-se um sistema dado por:

$$Au = r_u \tag{3.40}$$

onde a matriz A não sofre nenhuma alteração ao longo da simulação.

Essa propriedade é vantajosa na utilização de *solvers* diretos baseados em fatorização, como por exemplo LU e Cholesky, dado que o estágio de fatorização somente precisa ser computado uma única vez no início da simulação. Após o primeiro passo de tempo, a matriz decomposta (e pivoteada, onde necessário) é suficiente para que o sistema possa ser solucionado por substituição a cada novo passo.

No caso bidimensional, observa-se um sistema adicional desacoplado do anterior e dado por:

$$Av = r_v \tag{3.41}$$

onde a matriz A é exatamente a mesma do sistema descrito pela equação 3.40.

Desta forma o procedimento básico em cada passo de tempo após a inicialização se reduz a:

a) Calcular o vetor residual relativo a u, r_u.

b) Solucionar o sistema associado a u por substituição utilizando a matriz pré-fatorizada.

c) Calcular o vetor residual relativo a v, r_v.

d) Solucionar o sistema associado a v por substituição utilizando a matriz pré-fatorizada.

4 EXPERIMENTOS NUMÉRICOS

Neste capítulo são propostos experimentos numéricos para validar a implementação das soluções IMEX descritas no Capítulo 3. Com este fim, são utilizados casos teste em uma e duas dimensões já estudados na literatura, que servem como base de comparação para os novos esquemas numéricos. Algoritmos computacionais foram implementados para cada um dos cinco métodos propostos: FTCS, Crank-Nicolson, MCN-AX2+, AM2*-AX2*, e AI2*-AB3. Os códigos foram escritos na linguagem Fortran-90 para os dois casos (1D e 2D), utilizando a biblioteca LAPACK para a solução dos sistemas lineares onde necessário. Os resultados apresentados para os problemas propostos foram obtidos utilizando um *solver* baseado em fatorização LU; discussões sobre implementações de outros solvers são apresentadas na Seção 4.4.1.

Os incrementos temporais (Δt) e espaciais (Δx, Δy) nestes experimentos foram escolhidos de forma a garantir a estabilidade numérica dos métodos propostos, ao menos inicialmente. Com o aumento gradual da resolução espacial (i.e., menores valores de Δx e Δy) e/ou do coeficiente de viscosidade, os testes aproximam-se dos limites de certos métodos testados até eventualmente causar instabilidades numéricas.

Para resolver essa situação, um valor menor de Δt teria que ser escolhido; no entanto, o incremento temporal permanece constante. O objetivo desta estratégia é mostrar empiricamente que os métodos IMEX propostos permanecem estáveis em resoluções espaciais maiores que o método puramente explícito, dado um valor constante de Δt. Pelo mesmo raciocínio, se fosse mantida constante a resolução espacial, passos de tempo mais largos poderiam ser usados pelos métodos IMEX, diminuindo o número total de passos necessários para completar a simulação.

Menores incrementos temporais em teoria poderiam prover maior acurácia nos resultados; no entanto, Mesinger e Arakawa (1976) mostram que erros oriundos da discretização espacial possuem influência muito maior do que aqueles provenientes da discretização temporal, de forma que refinamentos na grade geram ganhos muito maiores na acurácia do método combinado do que a diminuição do valor de Δt.

4.1 Problema unidimemsional baseado em Nielsen et al. (2014)

Nielsen et al. (2014) mostram que uma solução exata para a equação de Burgers em uma dimensão, representando uma onda que se propaga da esquerda para a direita

conforme o tempo passa, é dada por:

$$u(x,t) = 1 - \tanh\left(\frac{x-t}{2\nu}\right) \tag{4.1}$$

O estudo realizado pelos autores considera um domínio espacial limitado por $x \in [-10, 10]$, com condições iniciais dadas pela Equação 4.1 em $t = 0$ e condições de contorno obtidas da mesma equação a cada passo de tempo discreto $n\Delta t$.

O primeiro caso teste baseado nesse estudo, aqui tratado como *baixa viscosidade*, utiliza um coeficiente de viscosidade $\nu = 0.0625$. O problema é integrado até $t = 5s$, utilizando um passo de tempo $\Delta t = 10^{-3}s$ (ou seja, a simulação é concluída após 5000 passos). A Tabela 4.1 apresenta o erro, calculado pela norma L^1 de u ao final da simulação, para cada um dos métodos propostos com variadas resoluções espaciais (identificadas pelo número de pontos ao longo de x). Nota-se que o aumento da resolução causa instabilidade no método explícito, e em $n_x = 2000$ o método se torna completamente inutilizável.

Tabela 4.1 - Erro da solução de cada método para o caso unidimensional com baixa viscosidade.

n_x	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
100	2.23521 E-02	2.19635 E-02	2.19641 E-02	2.19643 E-02	2.19647 E-02
125	1.14021 E-02	1.11344 E-02	1.11350 E-02	1.11353 E-02	1.11360 E-02
250	1.79243 E-03	1.72960 E-03	1.72952 E-03	1.72961 E-03	1.72987 E-03
500	4.34950 E-04	3.78677 E-04	3.78571 E-04	3.78631 E-04	3.78789 E-04
1000	1.53812 E-04	9.17426 E-05	9.16560 E-05	9.17135 E-05	9.18667 E-05
2000	N/A	2.28338 E-05	2.27288 E-05	2.27855 E-05	2.29382 E-05

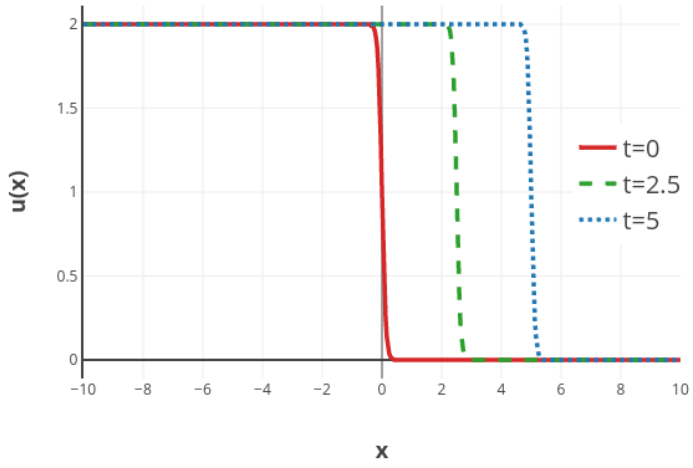
Uma análise empírica da taxa de convergência, realizada por meio da definição Big O proposta por Burden e Faires (2001) a partir dos erros previamente calculados, mostra que todos os métodos atingem convergência de segunda ordem conforme o esperado. Os resultados dessa análise são apresentados na Tabela 4.2, e confirmam a instabilidade do método explícito já quando $n_x = 1000$ (notável pela repentina discrepância na taxa de convergência).

Tabela 4.2 - Taxa de convergência da solução para u de cada método no caso unidimensional baseado em Nielsen et al. (2014) com baixa viscosidade.

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
125	3,017	3,044	3,044	3,044	3,044
250	2,669	2,687	2,687	2,687	2,686
500	2,043	2,191	2,192	2,192	2,191
1000	1,500	2,045	2,046	2,046	2,044
2000	N/A	2,006	2,012	2,009	2,002

A Figura 4.1 mostra que a solução obtida pelo esquema AI2*-AB3 em três diferentes estágios da integração numérica de *baixa viscosidade* ($t = 0$, $t = 2.5$, $t = 5s$), utilizando uma grade com 500 pontos, é consistente com os resultados obtidos por Nielsen et al. (2014) utilizando métodos Runge Kutta IMEX.

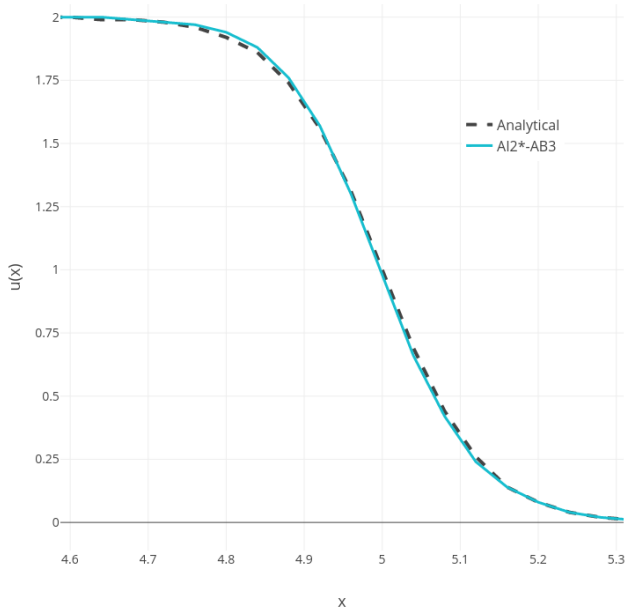
Figura 4.1 - Solução do caso unidimensional com baixa viscosidade em três passos de tempo (início, meio, e fim da simulação).



Fonte: Produção do autor.

Na Figura 4.2, a frente da onda é examinada de perto, mostrando que a solução IMEX (novamente obtida pelo esquema AI2*-AB3) praticamente se sobrepõe à solução analítica.

Figura 4.2 - Comparação entre as frentes de onda obtidas pela solução analítica e AI2*-AB3.



Fonte: Produção do autor.

Para o segundo caso teste, denominado *alta viscosidade*, são mantidas as mesmas condições iniciais e de contorno, assim como o passo de tempo $\Delta t = 10^{-3}s$. No entanto, utiliza-se um coeficiente de viscosidade mais alto, $\nu = 0.5$, para verificar sua influência na estabilidade da simulação. Os erros são apresentados na Tabela 4.3.

Tabela 4.3 - Erro da solução de cada método para o caso unidimensional com alta viscosidade.

n_x	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
100	1.14792 E-03	1.10247 E-03	1.10111 E-03	1.10112 E-03	1.10114 E-03
125	7.51261 E-04	7.01639 E-04	7.00076 E-04	7.00083 E-04	7.00102 E-04
250	2.29472 E-04	1.76976 E-04	1.75016 E-04	1.75023 E-04	1.75044 E-04
500	1.04186 E-04	4.59543 E-05	4.38034 E-05	4.38110 E-05	4.38312 E-05
1000	N/A	1.31947 E-05	1.09562 E-05	1.09637 E-05	1.09838 E-05
2000	N/A	5.01417 E-06	2.73723 E-06	2.74474 E-06	2.76501 E-06

As taxas de convergência são apresentadas na Tabela 4.4, e novamente nota-se que o método explícito rapidamente perde sua estabilidade mesmo com valores pequenos de n_x; os métodos IMEX, no entanto, mantém-se estáveis em todas as resoluções testadas. Nota-se que para $n_x > 500$, o método de Crank-Nicolson começa a apresentar taxas de convergência menores do que o esperado, independentemente de sua estabilidade incondicional. Uma possível explicação é que a precisão adotada para o corte nas iterações do método de Newton é muito baixa, afetando a acurácia da implementação.

Tabela 4.4 - Taxa de convergência da solução para u de cada método no caso unidimensional baseado em Nielsen et al. (2014) com alta viscosidade.

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
125	1,900	2,025	2,030	2,030	2,030
250	1,711	1,987	2,000	2,000	2,000
500	1,139	1,945	1,998	1,998	1,998
1000	N/A	1,800	1,999	1,999	1,997
2000	N/A	1,396	2,001	1,998	1,990

De forma geral, o experimento mostra que os três esquemas IMEX propostos são capaz de simular o problema com acurácia comparável à de um esquema puramente implícito, com a vantagem de o termo não linear ser tratado explicitamente; além disso, os novos métodos se mantém estáveis em resoluções que o método explícito é incapaz de resolver.

4.2 Problema bidimensional baseado em Zhu et al. (2010)

Com base no trabalho de Fletcher (1983) sobre a geração de soluções exatas para a equação de Burgers em duas dimensões, Zhu et al. (2010) propõem um caso teste em que as soluções para u e v são dadas pelas Equações 4.2 e 4.3, respectivamente:

$$u(x, y, t) = \frac{3}{4} - \frac{1}{4(1 + e^{(-t-4x+4y)/(32\nu)})} \tag{4.2}$$

$$v(x, y, t) = \frac{3}{4} + \frac{1}{4(1 + e^{(-t-4x+4y)/(32\nu)})} \tag{4.3}$$

Neste estudo, condições iniciais para u e v são dadas pelas Equações 4.2 e 4.3, respectivamente, em $t = 0$ e as condições de contorno obtidas a partir das mesmas

equações a cada passo de tempo discreto $n\Delta t$. O domínio espacial do problema, Ω_z, é dado por:

$$\Omega_z = \{(x, y) : 0 \leqslant x \leqslant 1, \quad 0 \leqslant y \leqslant 1\} \tag{4.4}$$

e o coeficiente de viscosidade é dado por $\nu = \frac{1}{80}$. O problema é integrado até $t = 0.5s$, utilizando um passo de tempo $\Delta t = 10^{-4}s$ (ou seja, a simulação é concluída após 5000 passos). A tabela 4.5 apresenta o erro, calculado pela norma L^1 de u ao final da simulação, para cada um dos métodos propostos com variadas resoluções espaciais (identificadas pelo número de células na grade); a Tabela 4.6 apresenta a mesma métrica para v.

Tabela 4.5 - Erro da solução para u de cada método no caso bidimensional baseado em Zhu et al. (2010).

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
10x10	9.75542 E-04	9.74883 E-04	9.74884 E-04	9.74884 E-04	9.74884 E-04
20x20	2.38169 E-04	2.37644 E-04	2.37644 E-04	2.37644 E-04	2.37644 E-04
30x30	1.04259 E-04	1.03780 E-04	1.03780 E-04	1.03780 E-04	1.03780 E-04
40x40	5.86397 E-05	5.81134 E-05	5.81135 E-05	5.81136 E-05	5.81136 E-05
50x50	3.77086 E-05	3.71849 E-05	3.71849 E-05	3.71850 E-05	3.71850 E-05

Tabela 4.6 - Erro da solução para v de cada método no caso bidimensional baseado em Zhu et al. (2010).

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
10x10	9.75542 E-04	9.74883 E-04	9.74884 E-04	9.74884 E-04	9.74884 E-04
20x20	2.38169 E-04	2.37644 E-04	2.37644 E-04	2.37644 E-04	2.37644 E-04
30x30	1.04259 E-04	1.03780 E-04	1.03780 E-04	1.03780 E-04	1.03780 E-04
40x40	5.86397 E-05	5.81134 E-05	5.81135 E-05	5.81136 E-05	5.81136 E-05
50x50	3.77086 E-05	3.71849 E-05	3.71849 E-05	3.71850 E-05	3.71850 E-05

Deve-se notar que, dada a precisão adotada, as tabelas parecem idênticas. Essa ilusão é causada pela similaridade das condições iniciais para u e v, que torna a diferença entre os erros tão pequena que só se torna aparente após 12 a 14 casas decimais – E-16 a E-18, na precisão adotada. Na prática, a diferença é tão próxima ao zero

da máquina que para todos os propósitos os erros em u e v podem ser considerados iguais. Um fenômeno semelhante ocorre na comparação entre os três métodos IMEX propostos, embora em menor escala. Por exemplo, na Tabela 4.5, o erro do método AM2*-AX2* em uma grade de 50x50 células é 3.7185015241578159E-05, enquanto o método AI2*-AB3 apresenta erro de 3.7185088058818792E-05. Novamente, devido à precisão adotada, os erros são apresentados como iguais para fins de comparação.

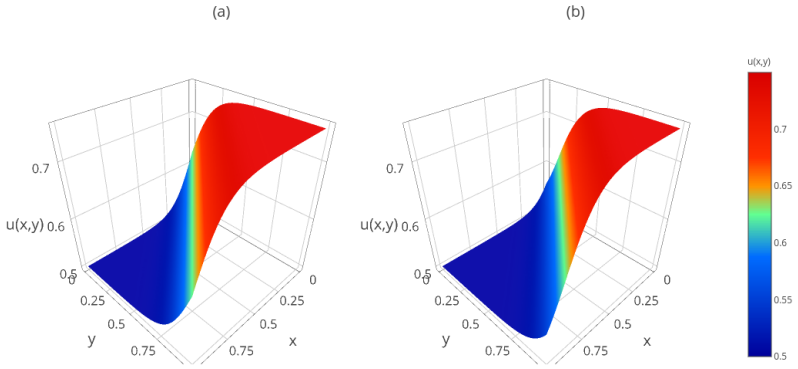
Todos os métodos IMEX propostos apresentam acurácia extremamente próxima à do método puramente implícito, e levemente superior à do método explícito estudado. A Tabela 4.7 apresenta a taxa de convergência de cada algoritmo para a solução de u, calculada a partir dos erros da Tabela 4.5. Os resultados da taxa de convergência de v são similares e portanto omitidos.

Tabela 4.7 - Taxa de convergência da solução para u de cada método no caso bidimensional baseado em Zhu et al. (2010).

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
20x20	2.034	2.036	2.036	2.036	2.036
30x30	2.037	2.043	2.043	2.043	2.043
40x40	2.000	2.015	2.015	2.015	2.015
50x50	1.978	2.001	2.000	2.000	2.000

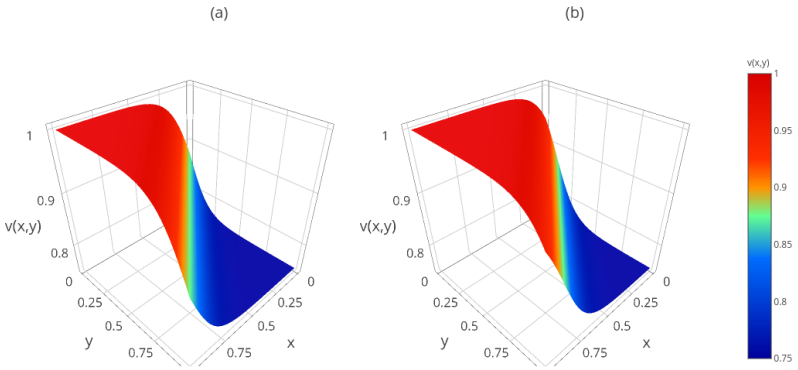
As Figuras 4.3 e 4.4 ilustram a evolução das soluções numéricas obtidas pelo método AI2*-AB3 para u e v, respectivamente, ao longo do tempo. Em ambas as figuras, o estágio (a) mostra as condições iniciais, na qual uma região quasi-descontínua nos campos de velocidade é propagada ao longo da simulação até atingir o instante final (b). Os resultados são consistentes com aqueles obtidos por Zhu et al. (2010) por meio de um método de decomposição Adomiana, confirmando assim a capacidade dos métodos de simular o problema proposto.

Figura 4.3 - Propagação da descontinuidade no campo de velocidade u na solução numérica nos estágios (a) inicial e (b) final da simulação baseada em Zhu et al. (2010).



Fonte: Produção do autor.

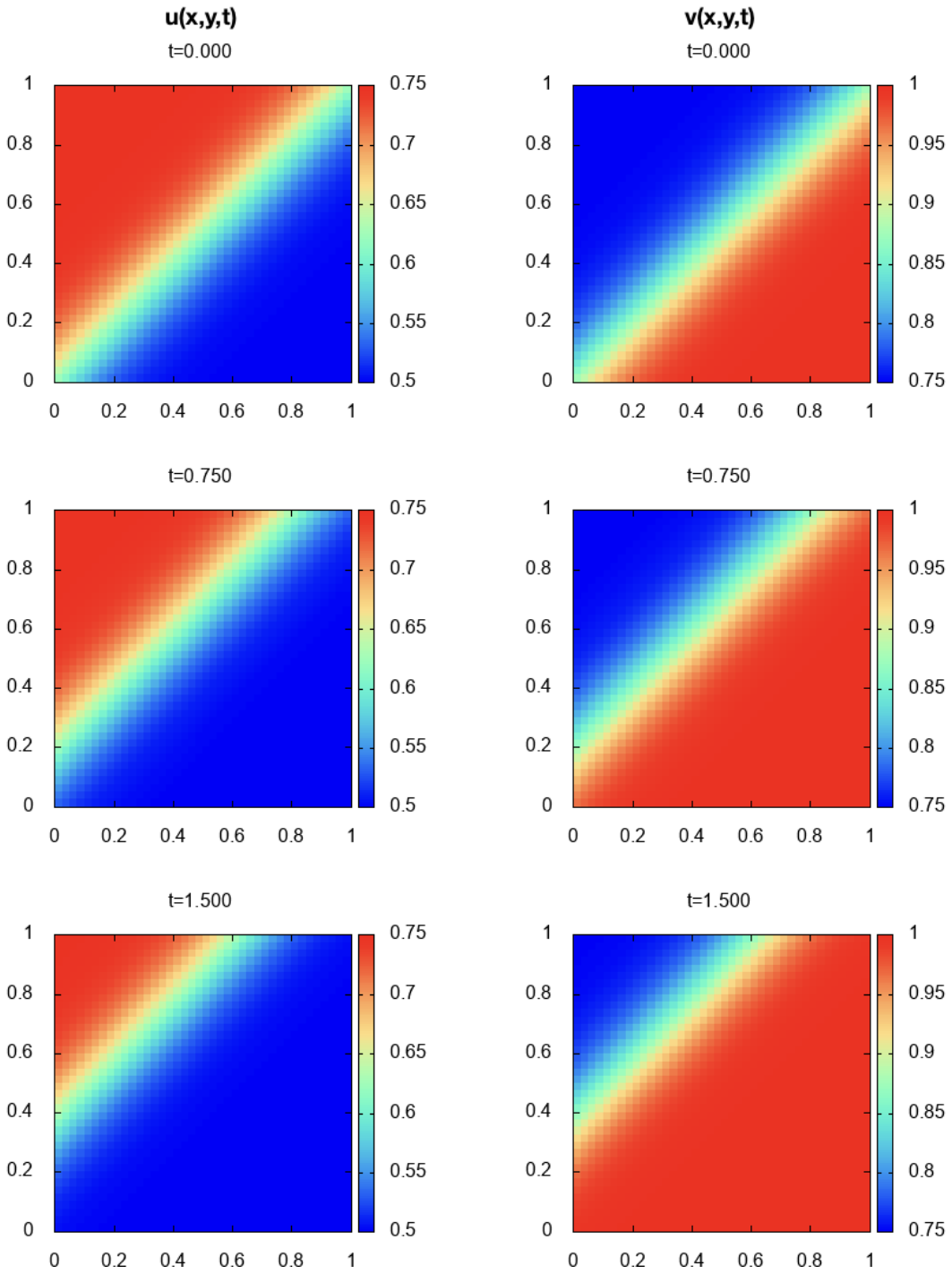
Figura 4.4 - Propagação da descontinuidade no campo de velocidade v na solução numérica nos estágios (a) inicial e (b) final da simulação baseada em Zhu et al. (2010).



Fonte: Produção do autor.

Para ilustrar mais claramente a evolução da região quasi-descontínua, foi realizada uma simulação mais longa, com um tempo total de $1.5s$ a partir das mesmas condições iniciais e com o mesmo passo de tempo. Os resultados são apresentados na Figura 4.5, na forma de um mapa de calor para u (esquerda) e v (direita) em três pontos distintos (início, meio e fim) da simulação.

Figura 4.5 - Evolução da solução de u e v ao longo de simulação mais longa baseada em Zhu et al. (2010).



Fonte: Produção do autor.

4.3 Problema bidimensional baseado em Kweyu et al. (2012)

Kweyu et al. (2012) mostram que um conjunto de soluções exatas para a equação de Burgers em duas dimensões pode ser obtido via transformada de Hopf-Cole (ver Apêndice A). Os autores propõem um caso de teste baseado no conjunto de parâmetros estabelecido na Tabela 4.8.

Tabela 4.8 - Parâmetros para a solução exata do primeiro caso teste de Kweyu et al. (2012).

a	b	c	d	A	B	C	D	E	α	β	γ
100	0	0	1	1	1	1	1	0	$\pi\sqrt{2}$	π	π

Desta forma, as soluções para u e v são reduzidas a:

$$u(x,y,t) = \nu \left[\frac{-2y - 2\pi e^{-2\nu\pi^2 t} \sin(\pi y)(\cos(\pi x) - \sin(\pi x))}{100 + xy + e^{-2\nu\pi^2 t} \sin(\pi y)(\cos(\pi x) + \sin(\pi x))} \right] \tag{4.5}$$

$$v(x,y,t) = \nu \left[\frac{-2x - 2\pi e^{-2\nu\pi^2 t} \cos(\pi y)(\cos(\pi x) + \sin(\pi x))}{100 + xy + e^{-2\nu\pi^2 t} \sin(\pi y)(\cos(\pi x) + \sin(\pi x))} \right] \tag{4.6}$$

O estudo utiliza como condições iniciais para u e v as equações 4.5 e 4.6, respectivamente, tomadas em $t = 0$. As condições de contorno são dadas pelas mesmas equações a cada passo de tempo discreto $n\Delta t$. O domínio espacial é dado por:

$$\Omega_k = \{(x,y) : 0 \leqslant x \leqslant 1, \quad 0 \leqslant y \leqslant 1\} \tag{4.7}$$

Dois casos de teste são propostos com base nesse estudo. O primeiro, baseado diretamente no trabalho de Kweyu et al. (2012), utiliza um baixo coeficiente de viscosidade ($\nu = 1/4000$). Este caso foi escolhido para validar diretamente os algoritmos bidimensionais implementados nesta dissertação, comparando o erro das soluções numéricas aos listados naquele trabalho para os métodos FTCS e Crank-Nicolson lá implementados. O problema é integrado até $t = 1s$, utilizando um passo de tempo $\Delta t = 10^{-3}s$ (ou seja, a simulação é concluída após 1000 passos). A tabela 4.9 apresenta o erro, calculado pela norma L^1 de u ao final da simulação, para cada um dos métodos propostos com variadas resoluções espaciais (identificadas pelo número de células na grade); a Tabela 4.10 apresenta a mesma métrica para v.

Os resultados obtidos por esta implementação dos métodos FTCS e Crank-Nicolson, apresentados nas Tabelas 4.9 e 4.10, são consistentes com os valores obtidos por Kweyu et al. (2012) para o caso teste proposto. Uma análise dos resultados revela que os três esquemas IMEX implementados reproduzem o problema com a mesma ordem de acurácia dos métodos anteriores para todas as resoluções testadas.

Tabela 4.9 - Erro da solução para u de cada método no caso bidimensional de baixa viscosidade baseado em Kweyu et al. (2012).

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
4x4	1.37267 E-09	1.37273 E-09	1.37273 E-09	1.37273 E-09	1.37273 E-09
8x8	4.58934 E-10	4.59020 E-10	4.59020 E-10	4.59020 E-10	4.59020 E-10
16x16	1.27003 E-10	1.27100 E-10	1.27100 E-10	1.27100 E-10	1.27101 E-10
32x32	3.29537 E-11	3.30561 E-11	3.30559 E-11	3.30559 E-11	3.30559 E-11
64x64	8.27575 E-12	8.37981 E-12	8.37960 E-12	8.37960 E-12	8.37961 E-12

Tabela 4.10 - Erro da solução para v de cada método no caso bidimensional de baixa viscosidade baseado em Kweyu et al. (2012).

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
4x4	7.73967 E-10	7.74001 E-10	7.74000 E-10	7.74000 E-10	7.74001 E-10
8x8	3.55949 E-10	3.56016 E-10	3.56016 E-10	3.56016 E-10	3.56016 E-10
16x16	1.10927 E-10	1.11013 E-10	1.11013 E-10	1.11013 E-10	1.11013 E-10
32x32	3.02421 E-11	3.03372 E-11	3.03370 E-11	3.03370 E-11	3.03371 E-11
64x64	7.73355 E-12	7.83221 E-12	7.83202 E-12	7.83202 E-12	7.83203 E-12

A análise das taxas de convergência (apresentadas nas Tabelas 4.11 e 4.12) confirmam a acurácia das implementações, coincidindo com os valores obtidos pelos autores do trabalho original.

Tabela 4.11 - Taxa de convergência da solução para u de cada método no caso bidimensional baseado em Kweyu et al. (2012) com baixa viscosidade.

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
8x8	1,581	1,580	1,580	1,580	1,580
16x16	1,853	1,853	1,853	1,853	1,853
32x32	1,946	1,943	1,943	1,943	1,943
64x64	1,993	1,980	1,980	1,980	1,980

Tabela 4.12 - Taxa de convergência da solução para v de cada método no caso bidimensional baseado em Kweyu et al. (2012) com baixa viscosidade.

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
8x8	1,121	1,120	1,120	1,120	1,120
16x16	1,682	1,681	1,681	1,681	1,681
32x32	1,875	1,872	1,872	1,872	1,872
64x64	1,967	1,954	1,954	1,954	1,954

Deve-se notar que o coeficiente de viscosidade escolhido pelos autores é extremamente pequeno – da ordem de 10^{-3} – e portanto a contribuição do termo difusivo $g(u)$ é pequena com relação à do termo convectivo, $f(u)$. Por este motivo, um novo experimento é proposto com base nas mesmas condições iniciais e de contorno, porém utilizando um coeficiente difusivo mais significativo dado por $\nu = 0.5$. O problema é novamente integrado até $t = 1s$, utilizando um passo de tempo $\Delta t = 10^{-3}s$ (ou seja, a simulação é concluída após 1000 passos). A Tabela 4.13 apresenta o erro, calculado pela norma L^1 de u ao final da simulação, para cada um dos métodos propostos com variadas resoluções espaciais (identificadas pelo número de células na grade); a Tabela 4.14 apresenta a mesma métrica para v.

Tabela 4.13 - Erro da solução para u de cada método no caso bidimensional de alta viscosidade baseado em Kweyu et al. (2012).

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
4x4	3.37083 E-007	3.78923 E-007	3.78788 E-007	3.78638 E-007	3.78238 E-007
8x8	5.20385 E-008	8.58507 E-008	8.57402 E-008	8.56140 E-008	8.52775 E-008
16x16	1.10676 E-008	2.09068 E-008	2.08018 E-008	2.06812 E-008	2.03598 E-008
32x32	N/A	5.15543 E-009	5.05180 E-009	4.93264 E-009	4.61497 E-009
64x64	N/A	1.24730 E-009	1.14401 E-009	1.02521 E-009	7.08456 E-010

Tabela 4.14 - Erro da solução para v de cada método no caso bidimensional de alta viscosidade baseado em Kweyu et al. (2012).

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
4x4	1.55049 E-008	1.72357 E-008	1.72329 E-008	1.72264 E-008	1.72090 E-008
8x8	2.94051 E-009	4.76637 E-009	4.76383 E-009	4.75695 E-009	4.73862 E-009
16x16	6.41358 E-010	1.21990 E-009	1.21743 E-009	1.21045 E-009	1.19182 E-009
32x32	N/A	3.03978 E-010	3.01532 E-010	2.94531 E-010	2.75867 E-010
64x64	N/A	7.37489 E-011	7.13001 E-011	6.42968 E-011	4.56236 E-011

O aumento da viscosidade, que também é um fator determinante nas condições de CFL do problema, causa instabilidade no método FTCS já em grades pequenas. Esse fenômeno pode ser claramente constatado no erro relativo à grade de 16x16 células, uma resolução muito menor que o limite observado em outros métodos. Incrementos à resolução confirmam a instabilidade do método, e a execução do algoritmo se torna inviável. Os esquemas IMEX testados são capazes de resolver o problema para as grades testadas; no entanto, nota-se que o esquema AI2*-AB3 começa a apresentar uma taxa de convergência maior do que a esperada nas grades de 32x32 células, e o esquema AM2*-AX2* nas grades de 64x64 células. O esquema MCN-AX2+ é o único cuja acurácia se mantém próxima à do método de Crank-Nicolson, mostrando-se mais robusto para este caso de teste. A análise se confirma com base nas taxas de convergência, apresentadas nas Tabelas 4.15 e 4.16.

Tabela 4.15 - Taxa de convergência da solução para u de cada método no caso bidimensional baseado em Kweyu et al. (2012) com alta viscosidade.

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
8x8	2,695	2,142	2,143	2,145	2,149
16x16	2,233	2,038	2,043	2,050	2,066
32x32	N/A	2,020	2,042	2,068	2,141
64x64	N/A	2,047	2,143	2,266	2,704

Tabela 4.16 - Taxa de convergência da solução para v de cada método no caso bidimensional baseado em Kweyu et al. (2012) com alta viscosidade.

Grade	FTCS	CN	MCN-AX2+	AM2*-AX2*	AI2*-AB3
8x8	2,399	1,854	1,855	1,857	1,861
16x16	2,197	1,966	1,968	1,974	1,991
32x32	N/A	2,005	2,013	2,039	2,111
64x64	N/A	2,043	2,080	2,196	2,596

4.4 Desempenho computacional da implementação proposta

Os algoritmos implementados foram também avaliados com relação ao seu desempenho computacional. Para tanto, o código foi instrumentado e dividido em duas seções principais: inicialização (incluindo o cálculo e a subsequente fatorização da matriz de coeficientes dos sistemas lineares associados aos esquemas IMEX) e integração temporal (restrita apenas ao tempo gasto com os passos de tempo do esquema escolhido). Métodos explícitos não gastam tempo significativo com a inicialização, e portanto esse estágio é apenas considerado (e discutido com mais detalhes na seção 4.4.1) para os métodos IMEX. Dada a implementação generalizada dos métodos de Durran-Blossey, os métodos IMEX são avaliados conjuntamente, e as rodadas consideram o esquema MCN-AX2+ como representativo de todas as implementações.

A Tabela 4.17 apresenta os tempos de execução (em segundos) para cada algoritmo na solução do problema de baixa viscosidade apresentado na Seção 4.3, cuja integração é composta de 1000 passos de tempo. As medidas apresentadas constituem

a média aritmética de 10 rodadas independentes do mesmo problema.

Tabela 4.17 - Tempo de execução (em segundos) dos algoritmos implementados para solução do problema de baixa viscosidade baseado em Kweyu et al. (2012).

Grade	FTCS	IMEX (LU)	Crank-Nicolson
4	0.00235	0.00352	0.12396
8	0.00441	0.00960	0.29586
16	0.00865	0.08364	8.39739
32	0.02011	2.25609	94.50982
64	0.05001	45.89848	2014.19368
128	0.14649	764.85504	33527.73682

Nota-se que o desempenho da implementação IMEX é extremamente fraco quando comparado ao da implementação FTCS, com o problema amplificando-se exponencialmente com o aumento da resolução, mas ainda assim apresentando desempenho muito superior ao do método implícito.

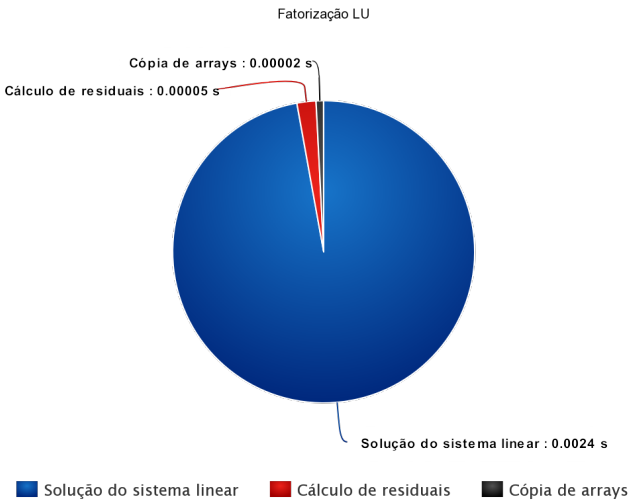
4.4.1 Desempenho de *solvers* para sistemas lineares

A solução dos sistemas de equações lineares para u e v é responsável pelo gargalo computacional em cada passo de tempo das implementações IMEX propostas. Sendo assim, a eficiência do *solver* escolhido para esse estágio do algoritmo é um ponto chave no desempenho dos métodos. Nesta dissertação são apresentados testes com dois *solvers* diferentes, ambos incluidos na biblioteca padrão LAPACK para cálculos de álgebra linear.

Para fins de consistência com a literatura existente, o primeiro *solver* testado é baseado em fatorização LU, a mesma utilizada nas implementações de Srivastava et al. (2011) e Kweyu et al. (2012). A rotina completa – que inclui o estágio de inicialização – associada a esse tipo de solução no LAPACK é denominada DSGESV, porém cada passo de tempo executa apenas sua componente DGETRS (com inicialização prévia via DGETRF). O tempo total de computação gasto na integração temporal foi apresentado na Tabela 4.17. A solução numérica em cada passo é constituída de três estágios: cálculo dos residuais do sistema, solução (por meio de substituição retroativa), e cópia de arrays para armazenamento de passos necessários no futuro (bem como descarte de passos que não serão utilizados novamente). Considerando

uma grade de 32x32 células, e o mesmo problema de baixa viscosidade, é claramente observável que o esforço computacional ocorre no estágio de solução do sistema (Figura 4.6).

Figura 4.6 - Tempo gasto por tarefa em um passo de tempo IMEX com solver LU.



Fonte: Produção do autor.

Uma outra opção para o problema estudado é a utilização de *solvers* baseados na decomposição de Cholesky. *Solvers* desse tipo são normalmente até duas vezes mais rápidos do que aqueles baseados em fatorização LU (PRESS et al., 1992); a implementação do LAPACK apresenta um desempenho superior ao *solver* LU da mesma biblioteca, reduzindo o tempo de execução conforme os resultados apresentados na Tabela 4.18.

Tabela 4.18 - Comparação do tempo de execução (em segundos) dos algoritmos IMEX com fatorização LU e Cholesky.

Grade	IMEX (LU)	IMEX (Cholesky)
4	0.00352	0.00405
8	0.00960	0.00950
16	0.08364	0.08290
32	2.25609	1.42639
64	45.89848	40.45639
128	764.85504	717.08838

A rotina completa – que inclui o estágio de inicialização – associada a esse tipo de solução no LAPACK é denominada DSPOSV, porém cada passo de tempo executa apenas sua componente DPOTRS (com inicialização prévia via DPOTRF). No entanto, o ganho de desempenho para o problema é bem menor do que o proposto pelos autores devido à execução incompleta do algoritmo. O estágio de fatorização da matriz A é realizado apenas uma vez, no início da execução, de forma que apenas trecho relativo à solução do sistema já fatorizado é executada em casa passo de tempo. A Tabela 4.19 apresenta o tempo gasto com a inicialização dos algoritmos, confirmando a maior velocidade da decomposição de Cholesky em comparação à LU, mas esse ganho torna-se irrelevante pelos motivos previamente citados.

Tabela 4.19 - Comparação do tempo de inicialização (em segundos) das decomposições LU e de Cholesky.

Grade	LU	Cholesky
4	0.00005	0.00003
8	0.00028	0.00005
16	0.00146	0.00138
32	0.04023	0.02449
64	2.38865	1.15225
128	125.46119	62.55341

A Figura 4.7 apresenta a divisão do tempo gasto em cada tarefa do passo de tempo com solver de Cholesky para o mesmo problema considerado na Figura 4.6. Deve-se notar, no entanto, que esta só pode ser utilizada em problemas cuja matriz de coeficientes é definida positiva (HIGHAM, 2009), ou seja, cujos autovalores são todos positivos. No caso das implementações propostas para a equação de Burgers, isso não gera restrição alguma, já que a matriz de coeficientes compartilhada pelos dois sistemas é compatível com esses requisitos; outros problemas podem não satisfazer as condições de uso deste *solver*.

Figura 4.7 - Tempo gasto por tarefa em um passo de tempo IMEX com solver Cholesky.

Fonte: Produção do autor.

A implementação final dos algoritmos propostos nesta dissertação (disponível no Apêndice B) substitui as rotinas LU pela fatorização de Cholesky, que oferece maior desempenho; no entanto, nota-se que certos parâmetros da tornaram-se obsoletos (em especial, o array de pivoteamento relacionado à decomposição LU não é mais utilizado). Esses parâmetros foram mantidos na transcrição, assim como as antigas rotinas LU (comentadas nas linhas 395 e 397), por questão de transparência com os resultados apresentados nos os experimentos numéricos.

5 CONCLUSÕES

Esta dissertação aplicou esquemas IMEX *multistep* baseados nos métodos de Adams, conforme propostos por Durran e Blossey (2012), à solução da equação de Burgers, tanto em sua forma unidimensional quanto em duas dimensões espaciais. A equação de Burgers é um modelo simplificado na dinâmica de fluidos, capaz de simular processos convectivos não lineares e processos difusivos viscosos. Implementações em Fortran-90 dos esquemas de Durran-Blossey foram comparadas a uma solução puramente explícita baseada no método de Euler progressivo (FTCS) e também a uma puramente implícita baseada no método de Crank-Nicolson, ambas já estudadas na literatura para o problema tratado.

Esquemas explícitos são facilmente implementáveis, mas não são convenientes na solução de problemas com escalas de tempo rápidas devido aos pequenos passos de tempo necessários para garantir estabilidade numérica – levando a um alto número de cálculos numéricos para completar simulações em altas resoluções espaciais. Esquemas implícitos, por sua vez, não são convenientes em problemas com escalas de tempo lenta que requerem a solução de sistemas não lineares a cada passo de tempo, devido ao alto custo computacional dos esquemas iterativos utilizados nesse estágio. Em princípio, esquemas numéricos podem ser úteis para resolver problemas que contenham as duas escalas, combinando as vantagens dos métodos explícitos e implícitos no tratamento de cada termo da EDP.

Experimentos numéricos baseados em soluções exatas da equação de Burgers em uma e duas dimensões foram escolhidos da literatura para validar a acurácia das soluções obtidas por meio dos métodos IMEX de Durran-Blossey. Os resultados mostram claramente que a acurácia de todos os esquemas IMEX implementados é semelhante à do método puramente implícito, mesmo com o tratamento explícito do termo não linear. Além disso, os novos métodos continuam estáveis em resoluções muito maiores do que o esquema explícito é capaz de resolver, consituindo uma alternativa viável para a solução de casos desse tipo. Em particular, o método MCN-AX2+ apresentou os melhores resultados entre os três métodos testados, mantendo-se estável durante todos os problemas estudados.

Os métodos IMEX demandaram maior esforço computacional do que o método explícito, considerando o mesmo passo de tempo em ambos os métodos. Se o modelo for linear, será necessário a solução de um sistema de equações algébricas, o que acarreta um maior esforço computacional em relação aos métodos explícitos trabalhando-se com as mesmas discretizações temporal. Desta forma, o método IMEX só se torna

computacionalmente competitivo se a discretização do tempo usada é maior do que o método explícito, que provavelmente estará acima do limite de estabilidade do método, como indicam a literatura e nossos experimentos numéricos. Se a equação for não linear, haverá um incremento da computação para cada passo de tempo. O ganho na computação só acontecerá, como já mencionado, utilizando-se uma discretização do tempo muito maior do que a usada no método explícito.

Os resultados desta tese foram obtidos usando *solvers* de bibliotecas numéricas sem o uso de técnicas para matrizes esparsas. Uma investigação que merece ser efetuada é desenvolver técnicas que fazem uso da esparcidade da matriz. Para matrizes de grande porte, é possível que métodos iterativos para a solução de sistemas lineares tornem-se mais eficientes do que os métodos diretos, como os da decomposição LU e de Cholesky aqui utilizados.

5.1 Trabalhos futuros

Sugere-se que trabalhos futuros visando melhorias nas técnicas aqui apresentadas investiguem os seguintes aspectos:

- Estudo comparativo com métodos RK IMEX (Runge-Kutta Implicit-Explicit), que são foco de pesquisas recentes na solução de problemas de dinâmica da atmosfera.
- Uso de técnicas para matrizes esparsas, incluindo estruturas de dados e solução dos sistemas lineares (IMEX) ou linearizados (Crank-Nicolson), com foco no desempenho computacional desse estágio do algoritmo. Outras bibliotecas de álgebra linear, como a MUMPS, podem ser analisadas comparativamente à LAPACK.
- Estudo de problemas mais computacionalmente intensivos na região explícita (i.e.: maior número de termos em escala lenta do que em escala rápida) que ocorrem na dinâmica de fluidos, avaliando assim o impacto de sistemas lineares menos significativos com relação à totalidade do problema.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, D.; PLETCHER, R. H.; TANNEHILL, J. C. **Computational fluid mechanics and heat transfer**. 3. ed. Boca Raton, FL, USA: CRC Press, 2012. 16, 17

ANDERSON JUNIOR, J. D. Discretization of partial differential equations. In: WENDT, J. F. (Ed.). **Computational fluid dynamics: an introduction**. Berlin: Springer, 1996. v. 1, p. 87–103. 7

ASCHER, U. M.; RUUTH, S. J.; SPITERI, R. J. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. **Applied Numerical Mathematics**, v. 25, n. 2–3, p. 151–167, 1997. 1, 13

ASCHER, U. M.; RUUTH, S. J.; WETTON, B. T. R. Implicit-explicit methods for time-dependent partial differential equations. **SIAM Journal on Numerical Analysis**, v. 32, n. 3, p. 797–823, 1995. 1, 12, 13, 14, 23

BAHADIR, A. R. A fully implicit finite-difference scheme for two-dimensional Burgers' equations. **Applied Mathematics and Computation**, v. 137, n. 1, p. 131–137, 2003. 27

BALDAUF, M. Stability analysis for linear discretisations of the advection equation with Runge-Kutta time integration. **Journal of Computational Physics**, v. 227, n. 13, p. 6638–6659, 2008. 2

BATEMAN, H. Some recent researches on the motion of fluids. **Monthly Weather Review**, v. 43, n. 4, p. 163–170, 1915. 21

BURDEN, R. L.; FAIRES, J. D. **Numerical analysis**. 7. ed. Boston: Brooks/Cole, 2001. 841 p. ISBN 9780534382162. 34

BURGERS, J. M. A mathematical model illustrating the theory of turbulence. In: MISES, R. von; KÁRMÁN, T. von (Ed.). **Advances in applied mechanics**. New York: Elsevier, 1948. v. 1, p. 171–199. 21

CANUTO, C.; HUSSAINI, A.; QUARTERONI, A.; ZANG, T. **Spectral methods in fluid dynamics**. Berlin: Springer-Verlag, 1987. 568 p. 12

CENTRO DE PREVISÃO DE TEMPO E ESTUDOS CLIMÁTICOS. **Brazilian developments on the Regional Atmospheric Modeling System**: About BRAMS. 2016. Disponível em: <http://brams.cptec.inpe.br/about/>. Acesso em: 07 mai. 2016. 3

CHAPRA, S. C.; CANALE, R. P. **Numerical methods for engineers**. 6. ed. New York, NY, USA: McGraw-Hill, 2010. 960 p. 12

COURANT, R.; FRIEDRICHS, K.; LEWY, H. Über die partiellen differenzengleichungen der mathematischen physik. **Mathematische Annalen**, v. 100, n. 1, p. 32–74, 1928. 1

CRANK, J.; NICOLSON, P. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. **Mathematical Proceedings of the Cambridge Philosophical Society**, v. 43, n. 1, p. 50–67, 1947. 25

DURRAN, D. R. **Numerical methods for fluid dynamics**: with applications to geophysics. 2. ed. New York: Springer-Verlag, 2010. 516 p. ISBN 9781441964113. 17, 18

DURRAN, D. R.; BLOSSEY, P. N. Implicit-explicit multistep methods for fast-wave-slow-wave problems. **Monthly Weather Review**, v. 140, n. 4, p. 1307–1325, 2012. 3, 13, 14, 15, 28, 51

EILBECK, J. C.; MCGUIRE, G. R. Numerical study of the regularized long-wave equation I: numerical methods. **Journal of Computational Physics**, v. 19, n. 1, p. 43–57, 1975. 18, 19

FLETCHER, C. A. J. Generating exact solutions of the two-dimensional Burgers' equations. **International Journal for Numerical Methods in Fluids**, v. 3, n. 3, p. 213–216, 1983. 21, 37

FREITAS, S. R.; PANETTA, J.; LONGO, K. M.; RODRIGUES, L. F.; MOREIRA, D. S.; ROSÁRIO, N. E.; DIAS, P. L. S.; DIAS, M. A. F. S.; SOUZA, E. P.; FREITAS, E. D.; LONGO, M.; FRASSONI, A.; FAZENDA, A. L.; SANTOS E SILVA, C. M.; PAVANI, C. A. B.; EIRAS, D.; FRANÇA, D. A.; MASSARU, D.; SILVA, F. B.; SANTOS, F. C.; PEREIRA, G.; CAMPONOGARA, G.; FERRADA, G. A.; VELHO, H. F. C.; MENEZES, I.; FREIRE, J. L.; ALONSO, M. F.; GÁCITA, M. S.; ZARZUR, M.; FONSECA, R. M.; LIMA, R. S.; SIQUEIRA, R. A.; BRAZ, R.; TOMITA, S.; OLIVEIRA, V.; MARTINS, L. D. The Brazilian developments on the Regional Atmospheric Modeling System (BRAMS 5.2): an integrated environmental model tuned for tropical areas. **Geoscientific Model Development**, v. 10, n. 1, p. 189–222, 2017. 2

GHALAMBAZ, M.; IZADPANAHI, E.; NOGHREHABADI, A.; CHAMKHA, A. Study of the boundary layer heat transfer of nanofluids over a stretching sheet:

Passive control of nanoparticles at the surface. **Canadian Journal of Physics**, v. 93, n. 7, p. 725–733, 2015. 2

HIGHAM, N. J. Cholesky factorization. **Wiley Interdisciplinary Reviews Computational Statistics**, v. 1, n. 2, p. 251–254, 9 2009. ISSN 1939-5108. 49

HUGHSON, M. C.; MOUNTS, J. S.; BELK, D. M. **An analysis of the viscous Burgers equation as modeled by the MacCormack method**. Eglin Air Force Base, Florida, 1987. 91p. Relatório técnico. Disponível em: <http://www.dtic.mil/dtic/tr/fulltext/u2/a182763.pdf>. Acesso em: 27 nov. 2017. 21

KADIOGLU, S. Y.; KNOLL, D. A. An IMEX method for the Euler equations that posses strong non-linear heat conduction and stiff source terms (radiation hydrodynamics). In: SCHULZ, H. (Ed.). **Hydrodynamics**: advanced topics. Rijeka, Croatia: InTech, 2011. p. 293–318. 12

KIM, J.; MOIN, P. Application of a fractional-step method to incompressible Navier-Stokes equations. **Journal of Computational Physics**, v. 59, n. 2, p. 308–323, 1985. 12

KWEYU, M. C.; MANYONGE, W. A.; KOROSS, A.; SSEMAGANDA, V. Numerical solutions of the Burgers' system in two dimensions under varied initial and boundary conditions. **Applied Mathematical Sciences**, v. 6, n. 113, p. 5603–5615, 2012. xv, xvi, xix, 22, 24, 42, 43, 44, 45, 46, 47, 59

LAX, P. D.; RICHTMYER, R. D. Survey of the stability of linear finite difference equations. **Communications on Pure and Applied Mathematics**, v. 9, n. 2, p. 267–293, 1956. 1

LEMIEUX, J. F.; KNOLL, D. A.; LOSCH, M.; GIRARD, C. A second-order accurate in time implicit-explicit (IMEX) integration scheme for sea ice dynamics. **Journal of Computational Physics**, v. 263, p. 375–392, 2014. 12

LIU, D.; WANG, Y. High order numerical solutions to convection diffusion equations with different approaches. **Journal of Applied & Computational Mathematics**, v. 4, n. 2, p. 208, 2015. 22

MESINGER, F.; ARAKAWA, A. **Numerical methods used in atmospheric models, volume I**. Geneva, Switzerland: Global Atmospheric Research Program World Meteorological Organization, 1976. 75 p. 9, 10, 11, 33

NIELSEN, T. B.; FISHER, T. C.; FRANKEL, S. H. High-order implicit-explicit multi-block time-stepping method for hyperbolic PDEs. In: AEROSPACE SCIENCES MEETING, 52, 2014, National Harbor. **Proceedings...** Reston: AIAA SciTech Forum, p. 2014-0770, 2014. xv, xix, 7, 12, 33, 35, 37

NOGHREHABADI, A.; IZADPANAHI, E.; GHALAMBAZ, M. Analyze of fluid flow and heat transfer of nanofluids over a stretching sheet near the extrusion slit. **Computers & Fluids**, v. 100, p. 227–236, 2014. 2

PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W. T.; FLANNERY, B. P. **Numerical recipes in C**: the art of scientific computing. 2. ed. New York: Cambridge University Press, 1992. 994 p. ISBN 0-521-43108-5. 48

RICHTMYER, R. D.; MORTON, K. W. **Difference methods for initial-value problems**. 2. ed. New York: Interscience, 1967. 405 p. ISBN 9780470720400. 18

SMITH III, L. C. **Finite element approximations of Burgers' equation with Robin's boundary conditions**. 1997. 68 p. Dissertação (Mestrado em Matemática) — Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1997. Disponível em: <https://vtechworks.lib.vt.edu/handle/10919/36958>. Acesso em: 22 jul. 2017. 22

SRIVASTAVA, V. K.; TAMSIR, M.; BHARDWAJ, U.; SANYASIRAJU, Y. V. S. S. Crank-Nicolson scheme for numerical solutions of two-dimensional coupled Burgers' equations. **International Journal of Scientific & Engineering Research**, v. 2, n. 5, p. 44–50, 2011. 21, 22, 25, 47

WANG, D.; RUUTH, S. J. Variable step-size implicit-explicit linear multistep methods for time-dependent partial differential equations. **Journal of Computational Mathematics**, v. 26, n. 6, p. 838–855, 2008. 7

WELLER, H.; LOCK, S. J.; WOOD, N. Runge-Kutta IMEX schemes for the horizontally explicit/vertically implicit (HEVI) solution of wave equations. **Journal of Computational Physics**, v. 252, p. 365–381, 2013. 2, 12, 13

WENDT, J. F. (Ed.). **Computational fluid dynamics**: an introduction. 3. ed. Berlin, Germany: Springer, 2009. 332 p. 8

WICKER, L. J.; SKAMAROC, W. C. Time-splitting methods for elastic models using forward time schemes. **Monthly Weather Review**, v. 130, n. 8, p. 2088–2097, 2002. 2

WICKER, L. J.; SKAMAROCK, W. A time-splitting scheme for the elastic equations incorporating second-order Runge-Kutta time differencing. **Monthly Weather Review**, v. 126, n. 7, p. 1992–1999, 1998. 2

ZARZUR, A. M.; CAMPOS VELHO, H. F.; FREITAS, S. R.; STEPHANY, S. Combined explicit and implicit methods for time integration in partial differential equations. In: THERMAL AND FLUIDS ENGINEERING CONFERENCE, 3, 2018, Fort Lauderdale. **Proceeding...** Connecticut: Begellhouse, p. 151–158, 2018. 4

ZARZUR, A. M.; CAMPOS VELHO, H. F.; STEPHANY, S.; FREITAS, S. R. IMEX schemes for time integration of Burgers' equation. In: IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING, XXXVIII, 2017, Florianópolis. **Proceedings...** Florianópolis: ABMEC, 2017. 4

ZARZUR, A. M.; STEPHANY, S.; FREITAS, S. R.; CAMPOS VELHO, H. F. A comparison of explicit, implicit, and IMEX time integration schemes for the two-dimensional Burgers' system. In: CONFERENCE OF COMPUTATIONAL INTERDISCIPLINARY SCIENCE, 4, 2016, São José dos Campos. **Proceedings...** São José dos Campos: PACIS, 2016. 3

_____. Numerical experiments for time integration of 2D Burgers' equations. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, XXXVII, 2017, São José dos Campos. **Proceedings...** São Carlos: SBMAC, 2018. 3

ZHU, H.; SHU, H.; DING, M. Numerical solutions of two-dimensional Burgers' equations by discrete Adomian decomposition method. **Computers & Mathematics with Applications**, v. 60, n. 3, p. 840–848, 2010. xiii, xv, xix, 37, 38, 39, 40, 41

APÊNDICE A - SOLUÇÃO EXATA DA EQUAÇÃO DE BURGERS BIDIMENSIONAL

Neste apêndice, será mostrada a solução analítica para as equações 3.5 e 3.6 por meio da transformação de Hopf-Cole e separação de variáveis, reproduzindo e iluminando o procedimento proposto por Kweyu et al. (2012). As soluções serão utilizadas para gerar conjuntos de condições iniciais e de contorno para a equação de Burgers, permitindo que os resultados numéricos possam ser comparados a uma solução exata.

Inicialmente, é feita uma linearização das equações ao relacionar uma função $\phi(x, y, t)$ a $u(x, y, t)$ e $v(x, y, t)$, de forma que

$$u = \frac{-2}{R}\frac{\phi_x}{\phi} \tag{A.1}$$

$$v = \frac{-2}{R}\frac{\phi_y}{\phi} \tag{A.2}$$

Para simplificar a notação, sejam $u = f_1(\phi)$ e $v = f_2(\phi)$. As derivadas de u e v com relação a x, y e t são encontradas e substituídas em 3.5 e 3.6, resultando em um novo sistema:

$$\begin{aligned} f_1'(\phi)\phi_t + f_1(\phi)f_1'(\phi)\phi_x + f_2(\phi)f_1'(\phi)\phi_y = \\ \frac{1}{R}\left(f_1(\phi)''\phi_x^2 + f_1'(\phi)\phi_{xx} + f_1(\phi)''\phi_y^2 + f_1'(\phi)\phi_{yy}\right) \end{aligned} \tag{A.3}$$

$$\begin{aligned} f_2'(\phi)\phi_t + f_1(\phi)f_2'(\phi)\phi_x + f_2(\phi)f_2'(\phi)\phi_y = \\ \frac{1}{R}\left(f_2(\phi)''\phi_x^2 + f_2'(\phi)\phi_{xx} + f_2(\phi)''\phi_y^2 + f_2'(\phi)\phi_{yy}\right) \end{aligned} \tag{A.4}$$

Os passos seguintes tratam da equação A.3; a solução para A.4 é similar e portanto não será repetida. Assume-se que ϕ é limitado, de forma que $f_1'(\phi)$ e $f_2'(\phi)$ são funções não-nulas. Dividindo a equação A.3 por $f_1'(\phi)$, obtém-se:

$$\phi_t + f_1(\phi)\phi_x + f_2(\phi)\phi_y = \frac{1}{R}\left(\frac{f_1(\phi)''\phi_x^2}{f_1'(\phi)} + \phi_{xx} + \frac{f_1(\phi)''\phi_y^2}{f_1(\phi)''\phi_x^2} + \phi_{yy}\right) \tag{A.5}$$

Considerando que as derivadas com relação a ϕ podem ser obtidas das equações A.1 e A.2 e substituídas em A.5, pode-se obter:

$$\phi_t = \frac{1}{R}(\phi_{xx} + \phi_{yy}) \tag{A.6}$$

A equação A.6 é linear e pode ser resolvida por separação de variáveis. Feito isso, a solução ϕ pode ser transformada de volta para u e v utilizando as equações A.1 e A.2. Busca-se uma solução geral com a forma

$$\phi(x, y, t) = a + bx + xy + dxy + X(x)Y(y)T(t) \tag{A.7}$$

na qual $\phi_1 = a+bx+cy+dxy$ representa a solução bilinear, adicionada para establizar a solução geral, e $\phi_2 = X(x)Y(y)T(t)$ representa a solução separável obtida a partir da equação transformada. A solução separável pode ser escrita como:

$$\phi_2 = W(x, y)T(t) \tag{A.8}$$

Por conveniência, essa solução é obtida separando-se primeiro espaço e tempo, e depois x de y. Substituindo a expressão A.8 em A.6, obtém-se:

$$WT' = \frac{1}{R}(W''_{xx}T + W''_{yy}T) \tag{A.9}$$

Rearranjando a equação A.9 e introduzino o operador laplaciano Δ, obtém-se:

$$R\frac{T'}{T} = \frac{\Delta W}{W} = -\alpha^2 \tag{A.10}$$

onde α^2 é a constante de separação e o sinal negativo é utilizado pois espera-se uma função que decai com o tempo. Assim, as equações separadas são:

$$T' + \frac{\alpha^2 T}{R} = 0 \tag{A.11}$$

$$\Delta W + \alpha^2 W = 0 \tag{A.12}$$

A solução para A.11 é dada por:

$$T(t) = Ae^{\frac{-\alpha^2 t}{R}} \tag{A.13}$$

Para solucionar A.12, as variáveis x e y são novamente separadas na função $W(x, y)$, resultando em:

$$X''Y + XY'' + \alpha^2 XY = 0 \tag{A.14}$$

Divindo a equação por XY, obtém-se:

$$\frac{X''}{X} = -\frac{Y''}{Y} - \alpha^2 = -\beta^2 \tag{A.15}$$

onde β^2 é a constante de separação para $W(x, y)$. Novamente, são obtidas duas equações separadas

$$X'' + \beta^2 X = 0 \tag{A.16}$$

$$Y'' + (\alpha^2 - \beta^2)Y = 0 \tag{A.17}$$

cujas soluções gerais são dadas por:

$$X(x) = Bsin(\beta x) + Ccos(\beta x) \tag{A.18}$$

$$Y(y) = Dsin(\gamma y) + Ecos(\gamma y) \tag{A.19}$$

onde $\gamma = \sqrt{(\alpha^2 - \beta^2)}$. Assim é obtida a solução geral para a equação A.7:

$$\begin{aligned} \phi(x, y, t) = a + bx + cy + dxy + \\ Ae^{\frac{-\alpha^2 t}{R}}[Bsin(\beta x) + Ccos(\beta x)][Dsin(\gamma y) + Ecos(\gamma y)] \end{aligned} \tag{A.20}$$

A solução de $\phi(x, y, t)$ é agora transformada de volta para u e v, conforme as equações A.1 e A.2, resultando nas soluções analíticas generalizadas para o sistema bidimensional de equações de Burgers:

$$u(x,y,t) = \frac{-2[b + dy + \beta A e^{\frac{-\alpha^2 t}{R}}(Bcos\beta x - Csin\beta x)(Dsin\gamma y + Ecos\gamma y)]}{R[a + bx + cy + dxy + Ae^{\frac{-\alpha^2 t}{R}}(Bsin\beta x + Ccos\beta x)(Dsin\gamma y + Ecos\gamma y)]} \tag{A.21}$$

$$v(x,y,t) = \frac{-2[c + dx + \gamma A e^{\frac{-\alpha^2 t}{R}}(Bsin\beta x + Ccos\beta x)(Dcos\gamma y - Esin\gamma y)]}{R[a + bx + cy + dxy + Ae^{\frac{-\alpha^2 t}{R}}(Bsin\beta x + Ccos\beta x)(Dsin\gamma y + Ecos\gamma y)]} \tag{A.22}$$

As equações A.21 e A.22 podem então ser utilizadas para gerar diversos conjuntos de condições iniciais e de contorno, bem como soluções exatas para comparação dos resultados numéricos, através da variação dos diversos coeficientes.

APÊNDICE B - ROTINA GENERALIZADA PARA PASSO DE TEMPO DE DURRAN-BLOSSEY EM DUAS DIMENSÕES

```
!> Advances the numerical solution of the 2D Burgers' equation from time 'n'
!! to 'n+1' using a generic formulation of the Adams-family methods from
!! Durran & Blossey (2012).
!!
!! @param u_n       Values of u(x,y) for time step 'n'.
!! @param u_nm1     Values of u(x,y) for time step 'n-1'.
!! @param u_nm2     Values of u(x,y) for time step 'n-2'.
!! @param v_n       Values of v(x,y) for time step 'n'.
!! @param v_nm1     Values of v(x,y) for time step 'n-1'.
!! @param v_nm2     Values of v(x,y) for time step 'n-2'.
!! @param u_border Boundary conditions data structure for 'u'.
!! @param v_border Boundary conditions data structure for 'v'.
!! @param lu        Coefficients matrix for the linear system, factorized in
!!                  Cholesky form.
!! @param pivot     Pivoting indices for the linear system.
!! @param grid      Grid specifications.
!! @param coef      Commonly used numerical constants/coefficients.
!! @param n         Current timestep.
!! @param nu        Viscosity coefficient.
!! @param caseid    Test case identifier for the analytical solution
!!                  (used for boundary conditions).
!! @param ru        Pre-allocated array of residuals for the linear system
!!                  associated with 'u'.
!! @param rv        Pre-allocated array of residuals for the linear system
!!                  associated with 'v'.
subroutine db_timestep(u_n, u_nm1, u_nm2, v_n, v_nm1, v_nm2, &
                       u_border, v_border, lu, pivot,        &
                       grid, coef, n, nu, caseid, ru, rv)
  double precision, dimension(:,:), intent(inout) :: u_n
  double precision, dimension(:,:), intent(inout) :: u_nm1
  double precision, dimension(:,:), intent(inout) :: u_nm2
  double precision, dimension(:,:), intent(inout) :: v_n
  double precision, dimension(:,:), intent(inout) :: v_nm1
  double precision, dimension(:,:), intent(inout) :: v_nm2
  type(boundary_conditions),        intent(inout) :: u_border
  type(boundary_conditions),        intent(inout) :: v_border
  double precision, dimension(:,:), intent(in)    :: lu
  integer,          dimension(:),   intent(in)    :: pivot
  type(domain_2d),                  intent(in)    :: grid
  type(db_coefficients),            intent(in)    :: coef
  integer,                          intent(in)    :: n
  double precision,                 intent(in)    :: nu
  integer,                          intent(in)    :: caseid
  ! The 'ru' and 'rv' arrays store the linear system results.
  ! They don't include the borders, hence the (nx-2)*(ny-2) size.
  double precision, dimension(:), intent(inout) :: ru
  double precision, dimension(:), intent(inout) :: rv

  integer                                       :: i, j, idx
  integer                                       :: dims, status
  real :: start, finish

  ! Boundary conditions for 'u' taken from the exact solution: ------------
  do j=1, grid%ny
```

```
55      u_border%west(j)  = u_exact(grid%x(1),grid%y(j),grid%t(n),nu,caseid)
56    end do
57
58    do j=1, grid%ny
59      u_border%east(j)  = u_exact(grid%x(grid%nx),grid%y(j),grid%t(n),nu,caseid)
60    end do
61
62    do i=1, grid%nx
63      u_border%south(i) = u_exact(grid%x(i),grid%y(1),grid%t(n),nu,caseid)
64    end do
65
66    do i=1, grid%nx
67      u_border%north(i) = u_exact(grid%x(i),grid%y(grid%ny),grid%t(n),nu,caseid)
68    end do
69
70    ! Boundary conditions for 'v' taken from the exact solution: ------------
71    do j=1, grid%ny
72      v_border%west(j)  = v_exact(grid%x(1),grid%y(j),grid%t(n),nu,caseid)
73    end do
74
75    do j=1, grid%ny
76      v_border%east(j)  = v_exact(grid%x(grid%nx),grid%y(j),grid%t(n),nu,caseid)
77    end do
78
79    do i=1, grid%nx
80      v_border%south(i) = v_exact(grid%x(i),grid%y(1),grid%t(n),nu,caseid)
81    end do
82
83    do i=1, grid%nx
84      v_border%north(i) = v_exact(grid%x(i),grid%y(grid%ny),grid%t(n),nu,caseid)
85    end do
86
87    ! The 2 linear systems to be solved are: ---------------------------------
88    ! A * u = ru                A * v = rv
89    ! We start by calculating 'ru' and rv'.
90    ! Residuals for 'u': -----------------------------------------------------
91    ! Special treatment of points adjacent to two borders.
92    ! Southwest corner.
93    i=2
94    j=2
95    idx = (i-2)*(grid%ny-2)+(j-1)
96    ru(idx) = u_n(i,j) &
97              + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
98              + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
99              + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
100             + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
101             + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
102             + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
103             + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
104             + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
105             + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
106             + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1)) &
107             - coef%a3 * u_border%west(j) - coef%a5 * u_border%south(i)
108   ! Northwest corner.
109   j=grid%ny-1
110   idx = (i-2)*(grid%ny-2)+(j-1)
111   ru(idx) = u_n(i,j) &
```

```
112             + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
113             + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
114             + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
115             + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
116             + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
117             + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
118             + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
119             + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
120             + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
121             + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1)) &
122             - coef%a3 * u_border%west(j) - coef%a4 * u_border%north(i)
123 ! Southeast corner.
124 i=grid%nx-1
125 j=2
126 idx = (i-2)*(grid%ny-2)+(j-1)
127 ru(idx) = u_n(i,j) &
128             + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
129             + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
130             + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
131             + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
132             + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
133             + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
134             + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
135             + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
136             + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
137             + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1)) &
138             - coef%a2 * u_border%east(j) - coef%a5 * u_border%south(i)
139 ! Northeast corner.
140 j=grid%ny-1
141 idx = (i-2)*(grid%ny-2)+(j-1)
142 ru(idx) = u_n(i,j) &
143             + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
144             + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
145             + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
146             + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
147             + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
148             + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
149             + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
150             + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
151             + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
152             + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1)) &
153             - coef%a2 * u_border%east(j) - coef%a4 * u_border%north(i)
154 ! Special treatment of points adjacent to one border.
155 ! West border.
156 i=2
157 do j=3, grid%ny-2
158   idx = (i-2)*(grid%ny-2)+(j-1)
159   ru(idx) = u_n(i,j) &
160               + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
161               + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
162               + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
163               + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
164               + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
165               + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
166               + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
167               + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
168               + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
```

```
169               + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1)) &
170               - coef%a3 * u_border%west(j)
171   end do
172   ! East border.
173   i=grid%nx-1
174   do j=3, grid%ny-2
175     idx = (i-2)*(grid%ny-2)+(j-1)
176     ru(idx) = u_n(i,j) &
177               + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
178               + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
179               + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
180               + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
181               + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
182               + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
183               + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
184               + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
185               + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
186               + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1)) &
187               - coef%a2 * u_border%east(j)
188   end do
189   ! South border.
190   j=2
191   do i=3, grid%nx-2
192     idx = (i-2)*(grid%ny-2)+(j-1)
193     ru(idx) = u_n(i,j) &
194               + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
195               + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
196               + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
197               + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
198               + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
199               + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
200               + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
201               + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
202               + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
203               + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1)) &
204               - coef%a5 * u_border%south(i)
205   end do
206   ! North border.
207   j=grid%ny-1
208   do i=3, grid%nx-2
209     idx = (i-2)*(grid%ny-2)+(j-1)
210     ru(idx) = u_n(i,j) &
211               + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
212               + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
213               + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
214               + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
215               + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
216               + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
217               + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
218               + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
219               + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
220               + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1)) &
221               - coef%a4 * u_border%north(i)
222   end do
223   ! Remaining points (not adjacent to any border).
224   do j=3, grid%ny-2
225     do i=3, grid%nx-2
```

```
226       idx = (i-2)*(grid%ny-2)+(j-1)
227       ru(idx) = u_n(i,j) &
228                 + coef%b2 *  (u_n(i,j) * (u_n(i+1,j) - u_n(i-1,j))) &
229                 + coef%b3 *  (v_n(i,j) * (u_n(i,j+1) - u_n(i,j-1))) &
230                 + coef%b4 *  (u_nm1(i,j) * (u_nm1(i+1,j) - u_nm1(i-1,j))) &
231                 + coef%b5 *  (v_nm1(i,j) * (u_nm1(i,j+1) - u_nm1(i,j-1))) &
232                 + coef%b6 *  (u_nm2(i,j) * (u_nm2(i+1,j) - u_nm2(i-1,j))) &
233                 + coef%b7 *  (v_nm2(i,j) * (u_nm2(i,j+1) - u_nm2(i,j-1))) &
234                 + coef%b8 *  (u_n(i+1,j) - 2. * u_n(i,j) + u_n(i-1,j))  &
235                 + coef%b9 *  (u_n(i,j+1) - 2. * u_n(i,j) + u_n(i,j-1))  &
236                 + coef%b10 * (u_nm1(i+1,j) - 2. * u_nm1(i,j) + u_nm1(i-1,j)) &
237                 + coef%b11 * (u_nm1(i,j+1) - 2. * u_nm1(i,j) + u_nm1(i,j-1))
238     end do
239   end do
240
241   ! Residuals for 'v': ------------------------------------------------------
242   ! Special treatment of points adjacent to two borders.
243   ! Southwest corner.
244   i=2
245   j=2
246   idx = (i-2)*(grid%ny-2)+(j-1)
247   rv(idx) = v_n(i,j) &
248             + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
249             + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
250             + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
251             + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
252             + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
253             + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
254             + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
255             + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
256             + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
257             + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1)) &
258             - coef%a3 * v_border%west(j) - coef%a5 * v_border%south(i)
259   ! Northwest corner.
260   j=grid%ny-1
261   idx = (i-2)*(grid%ny-2)+(j-1)
262   rv(idx) = v_n(i,j) &
263             + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
264             + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
265             + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
266             + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
267             + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
268             + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
269             + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
270             + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
271             + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
272             + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1)) &
273             - coef%a3 * v_border%west(j) - coef%a4 * v_border%north(i)
274   ! Southeast corner.
275   i=grid%nx-1
276   j=2
277   idx = (i-2)*(grid%ny-2)+(j-1)
278   rv(idx) = v_n(i,j) &
279             + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
280             + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
281             + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
282             + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
```

```
283             + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
284             + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
285             + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
286             + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
287             + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
288             + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1)) &
289             - coef%a2 * v_border%east(j) - coef%a5 * v_border%south(i)
290   ! Northeast corner.
291   j=grid%ny-1
292   idx = (i-2)*(grid%ny-2)+(j-1)
293   rv(idx) = v_n(i,j) &
294             + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
295             + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
296             + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
297             + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
298             + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
299             + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
300             + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
301             + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
302             + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
303             + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1)) &
304             - coef%a2 * v_border%east(j) - coef%a4 * v_border%north(i)
305   ! Special treatment of points adjacent to one border.
306   ! West border.
307   i=2
308   do j=3, grid%ny-2
309     idx = (i-2)*(grid%ny-2)+(j-1)
310     rv(idx) = v_n(i,j) &
311               + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
312               + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
313               + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
314               + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
315               + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
316               + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
317               + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
318               + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
319               + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
320               + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1)) &
321               - coef%a3 * v_border%west(j)
322   end do
323   ! East border.
324   i=grid%nx-1
325   do j=3, grid%ny-2
326     idx = (i-2)*(grid%ny-2)+(j-1)
327     rv(idx) = v_n(i,j) &
328               + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
329               + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
330               + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
331               + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
332               + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
333               + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
334               + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
335               + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
336               + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
337               + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1)) &
338               - coef%a2 * v_border%east(j)
339   end do
```

```
340   ! South border.
341   j=2
342   do i=3, grid%nx-2
343     idx = (i-2)*(grid%ny-2)+(j-1)
344     rv(idx) = v_n(i,j) &
345               + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
346               + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
347               + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
348               + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
349               + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
350               + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
351               + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
352               + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
353               + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
354               + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1)) &
355               - coef%a5 * v_border%south(i)
356   end do
357   ! North border.
358   j=grid%ny-1
359   do i=3, grid%nx-2
360     idx = (i-2)*(grid%ny-2)+(j-1)
361     rv(idx) = v_n(i,j) &
362               + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
363               + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
364               + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
365               + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
366               + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
367               + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
368               + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
369               + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
370               + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
371               + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1)) &
372               - coef%a4 * v_border%north(i)
373   end do
374   ! Remaining points (not adjacent to any border).
375   do j=3, grid%ny-2
376     do i=3, grid%nx-2
377       idx = (i-2)*(grid%ny-2)+(j-1)
378       rv(idx) = v_n(i,j) &
379                 + coef%b2 *  (u_n(i,j) * (v_n(i+1,j) - v_n(i-1,j))) &
380                 + coef%b3 *  (v_n(i,j) * (v_n(i,j+1) - v_n(i,j-1))) &
381                 + coef%b4 *  (u_nm1(i,j) * (v_nm1(i+1,j) - v_nm1(i-1,j))) &
382                 + coef%b5 *  (v_nm1(i,j) * (v_nm1(i,j+1) - v_nm1(i,j-1))) &
383                 + coef%b6 *  (u_nm2(i,j) * (v_nm2(i+1,j) - v_nm2(i-1,j))) &
384                 + coef%b7 *  (v_nm2(i,j) * (v_nm2(i,j+1) - v_nm2(i,j-1))) &
385                 + coef%b8 *  (v_n(i+1,j) - 2. * v_n(i,j) + v_n(i-1,j))  &
386                 + coef%b9 *  (v_n(i,j+1) - 2. * v_n(i,j) + v_n(i,j-1))  &
387                 + coef%b10 * (v_nm1(i+1,j) - 2. * v_nm1(i,j) + v_nm1(i-1,j)) &
388                 + coef%b11 * (v_nm1(i,j+1) - 2. * v_nm1(i,j) + v_nm1(i,j-1))
389     end do
390   end do
391
392   ! Solve the linear systems using the Cholesky matrix computed by 'dpotrf'. ---
393   ! For LU factorization, 'dgetrf' must be used in the initialization instead.
394   dims = (grid%nx - 2) * (grid%ny - 2)
395   !call dgetrs('No transpose', dims, 1, lu, dims, pivot, ru, dims, status) ! LU
396   call dpotrs( 'U', dims, 1, lu, dims, ru, dims, status ) ! Cholesky
```

```
397   !call dgetrs('No transpose', dims, 1, lu, dims, pivot, rv, dims, status) ! LU
398   call dpotrs( 'U', dims, 1, lu, dims, rv, dims, status ) ! Cholesky
399   if (status .ne. 0) then
400     stop
401   end if
402
403   ! Saves the results. ----------------------------------------------------
404   ! Time step 'n-1' becomes 'n-2'.
405   u_nm2(:,:) = u_nm1(:,:)
406   v_nm2(:,:) = v_nm1(:,:)
407   ! Time step 'n' becomes 'n-1'.
408   u_nm1(:,:) = u_n(:,:)
409   v_nm1(:,:) = v_n(:,:)
410   ! Time step 'n+1' becomes 'n'.
411   ! LAPACK stores the result of the linear system in 'ru' and 'rv'.
412   do j=1, grid%ny
413     u_n(1,j)        = u_border%west(j)
414   end do
415   do j=1, grid%ny
416     u_n(grid%nx,j) = u_border%east(j)
417   end do
418   do i=1, grid%nx
419     u_n(i,1)        = u_border%south(i)
420     u_n(i,grid%ny) = u_border%north(i)
421   end do
422   do j=2, grid%ny-1
423     do i=2, grid%nx-1
424       idx = (i-2)*(grid%ny-2)+(j-1)
425       u_n(i,j) = ru(idx)
426     end do
427   end do
428
429   do j=1, grid%ny
430     v_n(1,j)        = v_border%west(j)
431   end do
432   do j=1, grid%ny
433     v_n(grid%nx,j) = v_border%east(j)
434   end do
435   do i=1, grid%nx
436     v_n(i,1)        = v_border%south(i)
437     v_n(i,grid%ny) = v_border%north(i)
438   end do
439   do j=2, grid%ny-1
440     do i=2, grid%nx-1
441       idx = (i-2)*(grid%ny-2)+(j-1)
442       v_n(i,j) = rv(idx)
443     end do
444   end do
445 end subroutine db_timestep
```

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programas de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. Aceitam-se tanto programas fonte quanto os executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.