View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositório Aberto da Universidade do Porto

Congresso de Métodos Numéricos em Engenharia 2011
Coimbra, 14 a 17 de Junho, 2011
© APMTAC, Portugal, 2011

IMPLEMENTAÇÃO MATLAB DO MÉTODO DOS ELEMENTOS FINITOS MÓVEIS PARA RESOLUÇÃO DE PROBLEMAS DE CONVECÇÃO-DIFUSÃO-REACÇÃO

Rui J. Robalo[1], Rui M. Almeida[1], Maria do Carmo Coimbra[2], Alírio E. Rodrigues[2]

1:Departamento de Matemática
Faculdade de Ciências
Universidade da Beira Interior
Rua Marquês D´Ávila e Bolama, 6200-001 Covilhã, Portugal
e-mail: {rrobalo, ralmeida}@ubi.pt, web: http://www.cmatubi.ubi.pt

2: Laboratório de Processos de Separação e Reacção, Laboratório Associado LSRE/LCM
Departamento de Engenharia Química, Faculdade de Engenharia
Universidade do Porto
Rua Dr. Roberto Frias s/n, 4200-465 Porto
e-mail: {mcoimbra, arodrig}@fe.up.pt, web: http://www.lsre.fe.up.pt

Palavras-chave: Matlab, Elementos Finitos Móveis, Equações de Derivadas Parciais, Malhas Adaptativas, Fronteiras Móveis, Problemas de Convecção-Difusão-Reacção

Resumo. *No presente trabalho apresentamos uma aplicação computacional desenhada em Matlab para a resolução de sistemas de equações diferenciais com derivadas parciais, dependentes do tempo em domínios espaciais de dimensão 1. O algoritmo numérico baseia-se na formulação do método dos Elementos Finitos Móveis (MEFM) com funções de base não lineares e malhas adaptativas associadas a cada uma das variáveis dependentes. O principal objectivo desta nova implementação do MEFM é que possa ser utilizada de modo simples por um utilizador que não domine as técnicas da linguagem FORTRAN para resolver de modo eficiente uma grande variedade de problemas evolutivos, incluindo problemas com fronteiras móveis. Os exemplos numéricos que apresentamos permitem avaliar a robustez e performance deste código numérico escrito em Matlab e baseado no MEFM.*

1. INTRODUÇÃO

As equações diferenciais com derivadas parciais (EDPs) que descrevem problemas de convecção-difusão-reacção originam, com alguma frequência, soluções com frentes móveis abruptas. O MEFM [1, 2, 3, 4] é um método com malha variável, continuamente deformável, que permite simular eficientemente este tipo de sistemas físicos dinâmicos. Em cada instante o método determina não só a solução mas também a malha espacial que melhor a representa. Assim, o MEFM reposiciona automaticamente os nós no domínio espacial concentrando-os nas regiões onde a solução exibe grandes declives. A solução numérica de um sistema de EDPs através do MEFM é obtida em duas etapas distintas: a discretização espacial por intermédio de elementos finitos e consequente aproximação das derivadas espaciais e a integração no tempo do sistema de equações diferenciais ordinárias (EDOs) resultante.

Nos últimos anos o Matlab tem sido cada vez mais usado na simulação de modelos matemáticos em diversas áreas da investigação científica como ferramenta computacional de alto desempenho. Por exemplo, em [5] é disponibilizado um pacote de funções do Matlab que implementam o método dos elementos finitos para resolver problemas de elasticidade. Saucez [6], usa um integrador do Matlab para determinar a solução do sistema de EDOs obtido após discretizar a variável espacial através de diferenças finitas.

A fase de integração do sistema semi-discreto de EDOs, que no código original existente desenvolvido em FORTRAN era realizada recorrendo à rotina LSODIS, é na presente implementação do MEFM efectuada recorrendo ao integrador *ode15s* [7] do Matlab, que é apropriado para resolver sistemas *stiff* de EDOs. Foi desenvolvido um conjunto de funções que implementam o MEFM. O código foi escrito em Matlab permitindo a determinação da solução aproximada do problema em instantes pré-estabelecidos, a sua visualização e validação dos resultados, a um utilizador com conhecimentos básicos deste software.

Alguns problemas caracterizados por apresentarem soluções com frentes móveis abruptas foram usados para testar o código e os resultados obtidos demonstram a sua performance e flexibilidade.

2. MODELO GENÉRICO DO PROBLEMA

A aplicação computacional que propomos no presente trabalho usa o MEFM com aproximações polinomiais de grau superior [3], para determinar a solução numérica de problemas de convecção-difusão-reacção descritos por um sistema de EDPs parabólicas

$$\frac{\partial \boldsymbol{y}}{\partial t} = \boldsymbol{F}\,\frac{\partial^2 \boldsymbol{y}}{\partial x^2} + \boldsymbol{g}\,, \qquad x \in I,\ t \geq 0 \tag{1}$$

onde $\boldsymbol{y} = \boldsymbol{y}(x,t)$ é o vector das n variáveis dependentes, x e t são as variáveis independentes de espaço e tempo, respectivamente, a matriz $\boldsymbol{F}$ é diagonal tendo-se $F_{ii} = f_i\left(x,t,\boldsymbol{y},\partial \boldsymbol{y}/\partial x\right)$ e o vector $\boldsymbol{g}$ é tal que $g_i = g_i\left(x,t,\boldsymbol{y},\partial \boldsymbol{y}/\partial x\right)$. Cada variável dependente está sujeita a uma condição inicial e nos extremos do domínio espacial considerou-se a hipótese de existirem condições de fronteira de Robin com coeficientes dependentes do tempo,

$$\alpha_m(t)\frac{\partial y_m}{\partial x} = \beta_m(t)\, y_m + \gamma_m(t), \qquad m = 1,...,n. \tag{2}$$

Se o problema tiver uma fronteira móvel então a sua posição $X_s(t)$ em cada instante deve ser determinada como parte da solução. Neste caso para completar o modelo assumimos que o movimento da fronteira móvel é definido pela equação geral

$$\frac{dX_s}{dt} = w\left(X_s, t, \mathbf{y}\big|_{X_s}, \frac{\partial \mathbf{y}}{\partial x}\Big|_{X_s} \right), \qquad x \in I(t),\ t \geq 0. \tag{3}$$

3. FORMULAÇÃO DO MÉTODO

Para se desenvolver o algoritmo numérico utilizou-se o MEFM com as seguintes características: a malha de elementos finitos associada a cada uma das variáveis dependentes, originada de uma partição inicial do seu domínio espacial, é independente das outras malhas e o comprimento de cada elemento finito vai variar no tempo por forma a que a solução seja convenientemente descrita; cada variável dependente do sistema (1) é aproximada por um polinómio interpolador de Lagrange de qualquer grau entre 1 e 11 (à escolha do utilizador) em cada um dos elementos finitos; os nós interiores em cada elemento finito são definidos em cada instante em que queremos calcular a solução numérica e a sua posição relativamente aos extremos é optimizada como no método da colocação ortogonal; cada função componente do vector solução numérica é suavizada na vizinhança dos nós de separação de elementos finitos através de polinómios cúbicos de Hermite; o domínio espacial é decomposto numa malha única, mais fina, constituída pelos pontos de separação de todas as partições de modo a assegurar o cálculo muito mais preciso dos integrais que envolvem as derivadas espaciais da aproximação Y_m de y_m, se o elemento finito em causa contiver frentes abruptas correspondentes a outras variáveis. A aproximação Y_m é, em cada instante, uma função contínua seccionalmente polinomial que depende das amplitudes nodais em todos os pontos de interpolação e da posição dos nós de separação entre elementos finitos. As equações gerais do MEFM obtêm-se minimizando a soma dos quadrados da norma L_2 dos resíduos das EDPs com respeito a cada uma das derivadas de primeira ordem dos parâmetros efectivos do método. O movimento nodal não é completamente livre porque são introduzidos termos de penalização na função objectivo para prevenir a ocorrência de singularidades. Deste modo cada uma das EDPs de (1) gera um sistema de EDOs. A partir daqui o procedimento para obter a solução numérica é automático, visto que é o integrador que ajusta o passo temporal e produz não só a solução mas também as malhas adaptativas que melhor a representam.
Uma descrição pormenorizada desta formulação do MEFM pode ser consultada em [4] e os detalhes da sua extensão para resolver problemas bifásicos com uma fronteira móvel encontram-se em [8].

4. DESENVOLVIMENTO DA APLICAÇÃO COMPUTACIONAL

O sistema semi-discreto de EDOs resultante da aplicação do MEFM ao modelo matemático dado pelas equações (1) e (2),

$$\boldsymbol{M}\left(t,\boldsymbol{Y}_v\right)\frac{d\boldsymbol{Y}_v}{dt}=\boldsymbol{f}\left(t,\boldsymbol{Y}_v\right) \tag{4}$$

onde $\boldsymbol{Y}_v$ é o vector das variáveis dependentes, constitui um problema de valor inicial e é resolvido através do integrador *ode15s*, recomendado para problemas *stiff* [7]. O utilizador deste integrador é obrigado a fornecer duas funções, que designaremos por ADDA e RES, onde define a matriz de massa $\boldsymbol{M}$ e o vector $\boldsymbol{f}$ do segundo membro do sistema a integrar, respectivamente. Uma terceira função que define a matriz jacobiana de $\boldsymbol{f}$ é opcional. Como a definição analítica desta é uma tarefa muito difícil de concretizar optámos por ser o próprio integrador a calculá-la numericamente. Foi ainda desenvolvido um conjunto de funções do Matlab que são usadas na ADDA e RES ou para a visualização dos resultados. A estrutura geral do código Matlab é a seguinte: i.) o programa principal onde é realizada a integração; ii.) um conjunto de funções que são dependentes do problema a resolver; iii.) um conjunto de funções gerais que executam automaticamente operações necessárias ao cálculo da solução aproximada. Vamos agora apontar os factos mais importantes de cada um destes conjuntos de funções começando por aquelas que mudam de um problema para outro e portanto têm que ser redefinidas pelo utilizador.

4.1. Funções fornecidas pelo utilizador

O utilizador deste código terá apenas que definir as condições iniciais das várias variáveis dependentes que figuram no sistema (4) e fornecer as rotinas com: i) as n funções f_i e g_i que dependem de x, t, $\boldsymbol{y}$ e $\partial\boldsymbol{y}/\partial x$, a que chamámos F1 e F2, respectivamente; ii) as condições de fronteira nos extremos esquerdo e direito do domínio espacial dadas pelas equações (2), que existirem para o modelo matemático que se pretende resolver. A estas funções chamámos BC1 e BC2, respectivamente. Na tabela seguinte apresenta-se o código Matlab destas quatro funções para a equação de Nagumo, um dos exemplos testados e que é descrito na próxima secção.

function valor=F1(M,x,t,y,dydx)
% M: identifica a M-ésima EDP
% valor: matriz com a mesma dimensão da entrada x
epsil = 0.001; valor = epsil*ones(size(x));
	function valor=F2(M,x,t,y,dydx)
	% M, valor: exactamente como na função F1
	delta = 1e-3; a = 0;
	valor = y*(1-y)*(y-a)/delta*ones(size(x));

function valor=BC1(M,y,tempo) % y: valor da M-ésima variável dependente do % sistema de EDPs no extremo ESQUERDO do % domínio espacial para t=tempo % valor: matriz com a mesma dimensão da entrada y valor = Sol_exacta(M,0,tempo)*ones(size(y));	function valor=BC2(M,y,tempo) % y: valor da M-ésima variável dependente do % sistema de EDPs no extremo DIREITO do % domínio espacial para t=tempo % valor: matriz com a mesma dimensão da entrada y valor = Sol_exacta(M,1,tempo)*ones(size(y));

Tabela 1. Funções F1.m, F2.m, BC1.m e BC2.m para a equação de Nagumo

Relativamente à *m*-ésima equação do sistema (1) a função BC1 poderá assumir as seguintes formas: valor=beta1(tempo)*y+gama1(tempo) se $\partial y_m/\partial x = \beta_m^1(t)\, y_m + \gamma_m^1(t)$ para $x = a$ (a é o extremo esquerdo de I); valor=gama1(tempo) se $y_m = \gamma_m^1(t)$ no referido extremo; função muda se não houver nenhuma condição de fronteira nesse extremo. Quanto à função BC2 assumirá formas análogas. As condições iniciais e os outros dados necessários para o problema que se deseja resolver são introduzidos através do ficheiro file_dados.m. Vamos agora descrever as variáveis que são definidas pelo utilizador neste ficheiro:

NTEQDP é o número total de EDPs do sistema (1) que deve ser inferior ou igual a 30.

XL1, **XL2** são os extremos esquerdo e direito do domínio espacial, respectivamente.

N0, **N1** indicam se XL1 e XL2, respectivamente, são ou não pontos de colocação. N0=0 se não há condição de fronteira em XL1 para todas as EDPs e se não queremos que XL1 seja nó de nenhum dos sistemas de elementos finitos, N0=1 em todos os outros casos.

INDIC é o número de pontos interiores de quadratura a usar para calcular qualquer integral onde intervenha o segundo membro do sistema (1), que deve ser inferior ou igual a 33 e maior ou igual ao número máximo de pontos interiores de um elemento finito.

NEF(M) é o número de elementos finitos associado à m-ésima EDP.

NCF1(M), **NCF2(M)** indicam o tipo de condição de fronteira existente em XL1 e XL2, respectivamente, para a m-ésima EDP. NCF1=1 se não há condição de fronteira no extremo inicial do domínio espacial, NCF1=2 se há uma condição de Dirichlet nesse extremo e NCF1=3 se a condição de fronteira em XL1 é de Neumann ou Robin.

NP(L) número de pontos interiores de colocação no L-ésimo elemento finito que deve ser no máximo 10.

CES é a matriz dos parâmetros necessários para o cálculo das forças internodais. CES(5,L,M) define o comprimento mínimo admissível para o L-ésimo elemento finito associado a y_m.

```
NTEQDP=1;       % 1 <= NTEQDP <= 30
XL1=0;XL2=1;    % XL1 < XL2, sendo [XL1,XL2] o domínio espacial
N0=1;N1=1;
INDIC=33;       % máx { máx{NP(L): L=1:NEF(M)}: M=1:NTEQDP } <= INDIC <= 33
matriz_1=[4, 2, 2];       % matriz_1(M,:)=[NEF(M), NCF1, NCF2], com 1 <= NEF(M) <= 30
%   matriz_2(M,1:NEF(M))= NP(1:NEF(M)) com dimensão NTEQDP por máx{NEF(M) : M= 1:NTEQDP}
%    Em cada linha, preencha com zeros as colunas desde NEF(M)+1 até à última
matriz_2=[8, 10, 10, 8];       % 0 <= NP(L) <= 10
[n_M,n_L]=size(matriz_2);  CES=zeros(6,n_L,n_M);
CES(1,:,:)=1e-5;  CES(3,:,:)=1e-3;  CES(4,:,:)=1e-2;  CES(5,:,:)=1e-5;  CES(6,:,:)=1e-5;
%   matriz_X(:,M)= X(L+1,M), abcissas iniciais dos nós de separação interiores
matriz_X(:,1)=[0.01; 0.03; 0.05];
%   matriz_Y(:,M)= yaux(L,M), ordenadas iniciais em TODOS os nós de separação
matriz_Y=Sol_exacta(1, [0; matriz_X(:,1); 1], 0);
tolA=1e-5; tolR=1e-5;
vector_t=0:.1:1;       % Definição dos tempos de saída de resultados para os PERFIS
vector_id=2;
vector_NCF1=1; vector_NCF2=1;
```

Tabela 2. Ficheiro file_dados.m para a equação de Nagumo

X(L,M) é a abcissa inicial do L-ésimo nó de separação de elementos finitos para a M-ésima EDP, onde L=2,3,…,NEF(M).
yaux(L,M) é a ordenada inicial em X(L,M), com L=1,2,3,…,NEF(M)+1.
tolA, **tolR** são as tolerâncias absoluta e relativa, respectivamente, a usar pelo integrador.
vector_t é o vector onde o utilizador define os tempos de saída de resultados específicos para traçar os perfis, sendo vector_t(1) o tempo inicial.
vector_id(M) indica como vão ser calculadas as ordenadas iniciais dos pontos interiores a cada elemento finito associado à M-ésima EDP. Se vector_id=0 as ordenadas são obtidas por interpolação linear, vector_id=1 indica que são definidas por uma função da variável espacial e são definidas recorrendo à solução exacta se vector_id=2.
vector_NCF1(M), **vector_NCF2(M)** indicam se há ou não uma condição de Dirichlet dependente do tempo nos extremos inicial e final, respectivamente, do domínio espacial para a M-ésima EDP. vector_NCF1=1 se há uma condição de fronteira do tipo referido e vector_NCF1=0 caso contrário.
O código completo de file_dados.m definido para a equação de Nagumo encontra-se na tabela 2 e na tabela 3 apresenta-se o código Matlab da função Sol_exacta.m que define a solução exacta do mesmo problema.

```
function u = Sol_exacta(M,x,t)
%   M: identifica a M-ésima EDP;   x: variável de espaço;   t: variável tempo
%   u: matriz com a mesma dimensão da entrada x
T=t*ones(size(x));
epsil=1e-3; delta=1e-3; a=0; c=(1-2*a)*sqrt(epsil/2/delta);
u=(1-tanh((x-c*T)/sqrt(8*epsil*delta)))/2;
```

Tabela 3. Função Sol_exacta.m para a equação de Nagumo

4.2. Programa principal e funções gerais da implementação do MEFM

Para além da ADDA.m e RES.m mencionadas no início desta secção, utilizamos ainda outra função a que chamámos ENTRAS.m que tem os seguintes objectivos: efectuar um controlo dos dados fornecidos pelo utilizador nomeadamente através do ficheiro file_dados.m; definir todos os parâmetros necessários para determinar os nós interiores a cada um dos elementos finitos, as derivadas de 1ª e 2ª ordem das funções básicas da interpolação de Lagrange, os pontos de quadratura e respectivos pesos e os valores das funções básicas de interpolação nos diferentes pontos de quadratura. No programa principal, antes de se chamar o *ode15s* é necessário definir os valores iniciais das variáveis do sistema de EDOs, designadas por $\boldsymbol{Y_v}$, que são os valores das ordenadas nos nós que não forem fixados por quaisquer condições de fronteira e as abcissas dos nós de separação entre elementos finitos. As ordenadas dos pontos interiores de colocação são obtidas por interpolação linear ou através de uma função da variável de espaço que pode ser a solução exacta, se existir. As amplitudes e as posições nodais encontram-se intercaladas no vector $\boldsymbol{Y_v}$, ordenadas de modo a conferir uma estrutura em banda à matriz dos coeficientes do sistema de EDOs. A mancha de elementos possivelmente não nulos da matriz $\boldsymbol{M}$ para a equação de Nagumo, considerando os dados da

tabela 2, apresenta a configuração próxima de uma matriz diagonal por blocos que a figura 1 ilustra. $\boldsymbol{M}$ é uma matriz esparsa e, neste exemplo em particular, com densidade máxima igual a 0.3288.

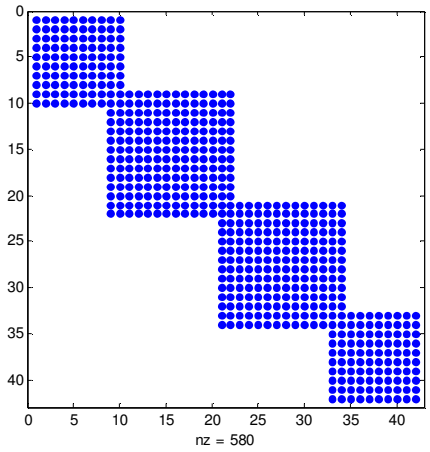

Figura 1. Blocos não-zeros da matriz $\boldsymbol{M}$ para o problema de Nagumo

A formulação do MEFM apresentada origina matrizes de massa $\boldsymbol{M}$ com grau de esparsidade elevado e fortemente dependentes de $\boldsymbol{Y_v}$. Assim, é fornecido ao integrador através das funções M_padrao.m e j_padrao.m, o padrão de esparsidade das matrizes $\partial(\boldsymbol{M}\,v)/\partial\boldsymbol{Y_v}$ e $\partial\boldsymbol{f}/\partial\boldsymbol{Y_v}$, respectivamente. Durante o ciclo de integração, após se atingir com sucesso cada um dos tempos de saída de resultados definidos em vector_t (ver tabela 2), faz-se a conversão das variáveis $\boldsymbol{Y_v}$ nas variáveis que definem as amplitudes e posições nodais e escreve-se a lista completa de resultados no ficheiro perfis.txt. São ainda armazenados os resultados numéricos obtidos em todos os passos intermédios da integração. Finda a integração, imediatamente a seguir se procede à visualização dos resultados. Esta tarefa é concretizada por cinco funções distintas. É traçado um gráfico com os perfis para os valores da variável tempo definidos pelo utilizador em vector_t, outro com o movimento nodal, outro ainda com as trajectórias dos nós de separação, mais um tridimensional que define a solução aproximada e finalmente um gráfico com as histórias num conjunto discreto de pontos do domínio espacial. Se o problema tiver solução exacta então, para cada um dos tempos de vector_t, é calculada a norma máxima do vector erro considerando todos os nós de colocação, incluindo os nós interiores. Durante todo este procedimento só por uma única vez é solicitada a intervenção do utilizador, no sentido de definir as concretizações da variável espacial onde pretende historiar os valores das variáveis dependentes do sistema (1). Por defeito são apresentadas apenas três histórias, em $x_k = x_{k-1} + 0.3\,c(I)$, $k = 1,2,3$, onde x_0 representa o extremo esquerdo de I e $c(I)$ o seu comprimento. As restantes rotinas executam tarefas específicas e são usadas pelas funções descritas anteriormente.

5. EXEMPLOS SIMULADOS

O MEFM foi aplicado na resolução de problemas de convecção-difusão-reacção. O código Matlab resultante da implementação do algoritmo numérico foi testado na simulação da equação de Nagumo, no modelo definido pelo sistema de equações de FitzHugh-Nagumo e num problema com uma fronteira móvel. Estes exemplos foram resolvidos no passado com códigos em linguagem FORTRAN. Os integrais que aparecem em cada uma das equações do sistema de EDOs são calculados recorrendo à integração numérica, através da quadratura de Lobatto, usando um número de pontos de quadratura suficiente para que o erro de truncatura seja nulo. Usámos valores de referência para as constantes das funções de penalização do movimento nodal e fixámos as tolerâncias do integrador em 1e-5. No entanto, de modo a efectuarmos a comparação com o código desenvolvido em FORTRAN, escolhemos para comprimento mínimo admitido de qualquer elemento finito 1e-7 e tolA=tolR=1e-8, no último problema apresentado. Realizámos todas as simulações num computador com um processador Intel Core2 Duo a 2.00 GHz com 3.00 GB de RAM usando Matlab.

5.1. Equação de Nagumo com solução analítica

O problema descrito em [9] consiste na equação

$$\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} + \frac{1}{\delta} u(1-u)(u-a), \quad 0 < x < 1 \text{ e } t > 0, \tag{5}$$

sujeita a condições de fronteira de Dirichlet e a condições iniciais de acordo com a solução exacta

$$u(x,t) = \frac{1}{2}\left[1 - \tanh\left(\frac{x - ct}{\sqrt{8\varepsilon\delta}}\right)\right], \; 0 \le x \le 1 \text{ e } t > 0, \text{ onde } c = (1-2a)\sqrt{\frac{\varepsilon}{2\delta}}. \tag{6}$$

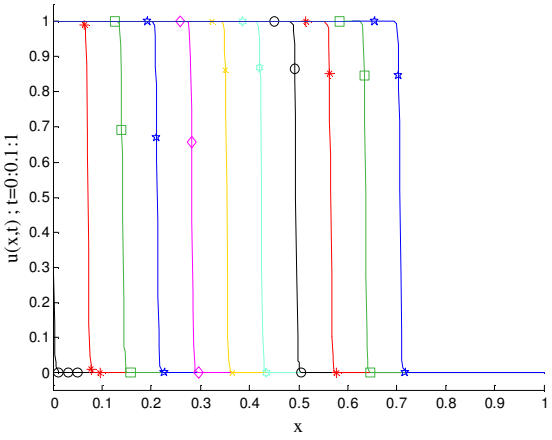

Figura 2. Equação de Nagumo com $\varepsilon = \delta = 0.001$

Inicialmente a solução desenvolve uma frente abrupta que mantém durante o seu deslocamento na direcção do extremo direito do domínio espacial. Na simulação usámos $\varepsilon = 0.001$, $\delta = \varepsilon$, $a = 0$ e os dados que constam na tabela 2. O tempo de CPU para completar todos os cálculos foi 106.5 segundos. Os resultados obtidos em diferentes instantes são apresentados na figura 2. A solução numérica produzida pela implementação Matlab do MEFM tem uma boa precisão, registando-se um erro máximo inferior a 1.4e-3 em *t*=0.6. As figuras 3 e 4 mostram a variação da posição dos nós ao longo do tempo e as trajectórias dos nós de separação, respectivamente. Podemos constatar o movimento dos nós de separação por forma a seguirem a frente abrupta e a melhor representarem a solução. Os valores da variável dependente foram historiados em três pontos do domínio espacial. Estas três histórias podem ser observadas na figura 5.

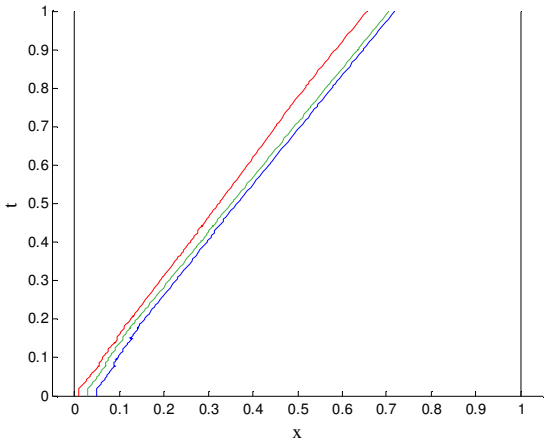

Figura 3. Movimento dos nós de separação

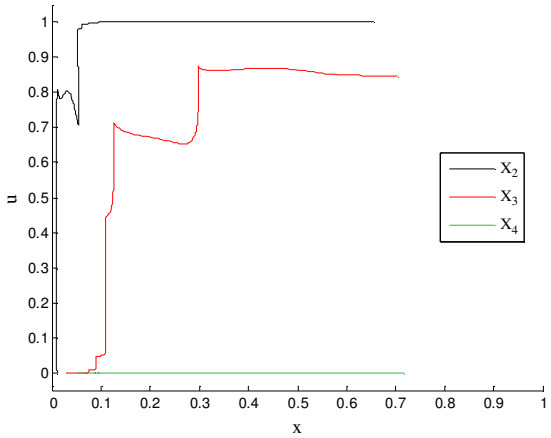

Figura 4. Trajectórias dos nós de separação

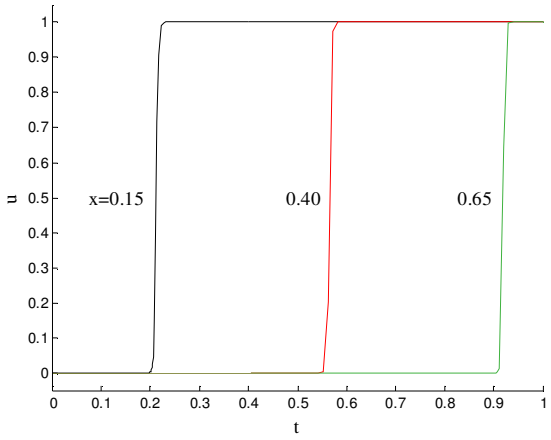

Figura 5. Histórias no problema de Nagumo

5.2. Problema de reacção-difusão da biologia

O modelo matemático deste problema, que foi estudado por Coimbra em [4] e Verwer em [10], é descrito na sua forma adimensional pelo sistema de equações (de FitzHugh-Nagumo),

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2} + u(1-u)(u-a) - v \\ \dfrac{\partial v}{\partial t} = b(u - c\,v) \end{cases}, \quad 0 < x < 120 \text{ e } t > 0, \tag{7}$$

onde u representa um potencial electroquímico e v é uma variável que permite que o sistema recupere o seu estado de repouso, pelas condições de fronteira,

$$\frac{\partial u}{\partial x}(0,t) = -\frac{I}{2} \quad \text{e} \quad \frac{\partial u}{\partial x}(120,t) = 0, \quad t \geq 0, \tag{8}$$

e condições iniciais nulas em todo o domínio espacial. Os valores considerados para os parâmetros do modelo são $a = 0.139$, $b = 0.008$, $c = 2.54$ e $I = 0.45$. A solução numérica foi calculada considerando uma discretização inicial do domínio espacial em 19 elementos finitos e usando aproximações polinomiais das variáveis dependentes de grau 4 em cada elemento finito da respectiva malha. As posições iniciais dos nós de separação móveis são iguais para ambas as variáveis dependentes e foram uniformemente distribuídas em]0,120[. Os valores das constantes de penalização são: $c_1 = c_3 = c_4 = 10^{-3}$, $c_2 = 0$, $c_5 = c_6 = 10^{-5}$ para o primeiro sistema de elementos finitos e c_3 e c_4 dez vezes maiores na malha associada a v, mantendo todos os outros inalterados. Nesta simulação o tempo de CPU foi 13 minutos e 11.8 segundos. A solução desenvolve repetidamente uma frente sob a forma de onda. Estas ondas surgem junto do extremo esquerdo do domínio espacial e deslocam-se para a direita. Logo que a onda

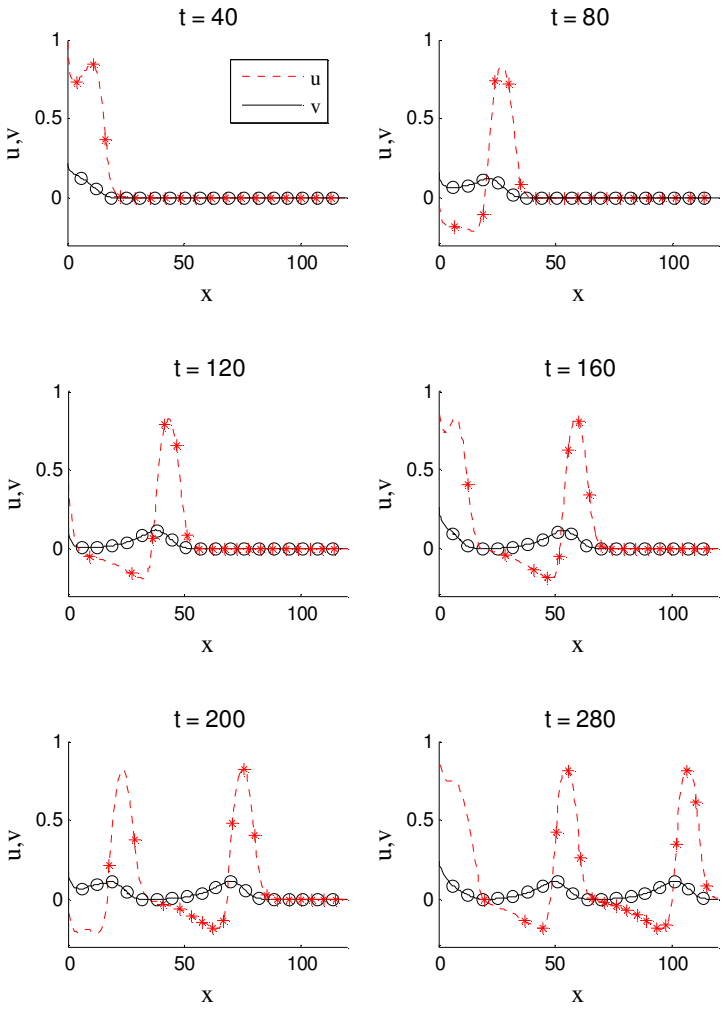

Figura 6. Movimento nodal e soluções em vários valores de *t*

recém-chegada percorra cerca de um quarto do seu trajecto uma nova onda irá formar-se. Assim, à medida que *t* aumenta há mais regiões onde a solução exibe grandes declives, sendo por isso necessária uma boa definição da malha para que seja convenientemente representada. As figuras 6 e 7 apresentam os perfis das soluções aproximadas e a distribuição dos valores da variável de recuperação do sistema no domínio espaço-tempo, respectivamente.

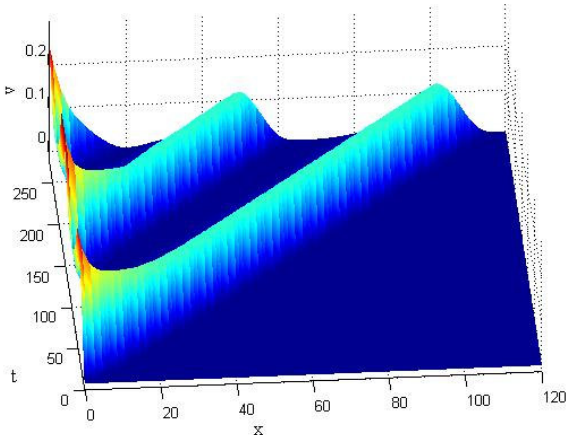

Figura 7. Distribuição dos valores de *v*

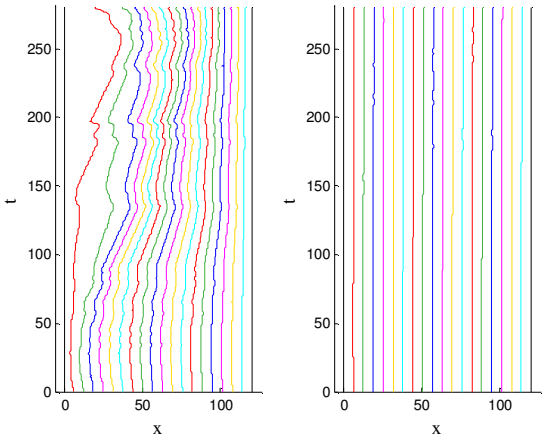

Figura 8. Movimento dos nós de separação associados *u* e *v*, respectivamente

A figura 8 mostra a evolução da posição dos nós de separação ao longo do tempo. É visível que os nós da primeira malha exibem actividade espacial, inflectindo o seu movimento lateral sempre que se revele necessário para que a solução seja adequadamente definida enquanto que os nós associados à segunda variável dependente apresentam variações ligeiras da sua posição. Na figura 9 visualizam-se histórias do potencial electroquímico. Podemos observar que os valores desta variável se repetem no tempo com uma determinada frequência.

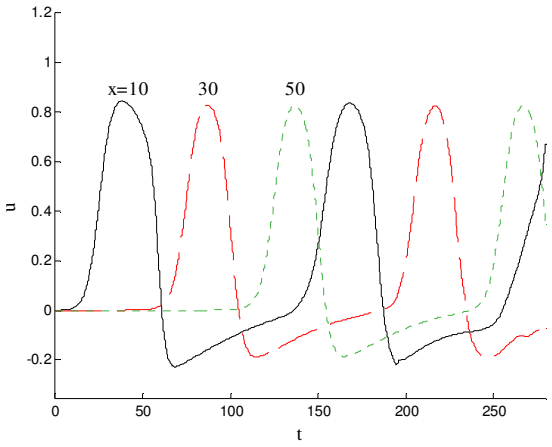

Figura 9. Histórias do potencial electroquímico

5.3. Problema com uma fronteira móvel

O último exemplo que apresentamos é o problema da fusão de um sólido formulado em [11]. O modelo adimensionalizado é descrito por

$$\begin{aligned} &\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < X_s(t), \quad 0 < t < 1 \\ &\frac{\partial u}{\partial x}(0,t) = -e^t \quad \text{e} \quad u(X_s(t),t) = 0, \quad t > 0, \\ &\frac{dX_s}{dt} = -\frac{\partial u}{\partial x}(X_s(t),t), \quad t > 0, \quad X_s(0) = 0 \end{aligned} \tag{9}$$

Pretende-se determinar a temperatura na fase líquida, $u(x,t)$, para $0 \le x \le X_s(t)$ e $t > 0$, e a posição da interface líquido/sólido $X_s(t)$, no instante t. Este problema tem solução exacta

$$u(x,t) = e^{t-x} - 1, \quad 0 \le x \le X_s(t) \quad \text{e} \quad X_s(t) = t, \quad t > 0. \tag{10}$$

O domínio espacial inicial é apenas um ponto, mas de modo a podermos aplicar o nosso algoritmo numérico vamos iniciar a simulação com uma fase líquida de comprimento 2×10^{-6}, $u(x,0) = 0$ para $0 \le x \le X_s(0)$ e definindo para a fase sólida um modelo análogo com solução nula. Os resultados numéricos foram obtidos usando aproximações locais de grau 3 em cada um dos 4 elementos finitos da fase líquida e um só elemento finito com 1 nó interior na fase sólida. A malha espacial inicial associada à fase líquida é uniforme.

t	$X_s(t)$-FOR	$X_s(t)$-M
0.1	0.1000018	0.1000020
0.2	0.2000018	0.2000019
0.3	0.3000018	0.3000019
0.4	0.4000022	0.4000023
0.5	0.5000032	0.5000031
0.6	0.6000050	0.6000046
0.7	0.7000079	0.7000072
0.8	0.8000121	0.8000110
0.9	0.9000178	0.9000166
Erro	< 1.8E-05	< 1.7E-05

Tabela 4. Comparação da posição da interface em diferentes instantes

Nas tabelas 4 e 5 comparam-se os resultados obtidos para a posição da fronteira móvel em vários instantes e para a distribuição dos valores da temperatura no instante t=0.5, respectivamente, com os existentes em [8] da simulação com o código desenvolvido em FORTRAN e com a solução exacta.

x/X_s	FORTRAN	MATLAB	Solução exacta
0.0	0.648725	0.648724	0.648721
0.1	0.568315	0.568314	0.568312
0.2	0.491828	0.491826	0.491825
0.3	0.419070	0.419069	0.419068
0.4	0.349861	0.349860	0.349859
0.5	0.284027	0.284026	0.284025
0.6	0.221404	0.221403	0.221403
0.7	0.161835	0.161835	0.161834
0.8	0.105171	0.105171	0.105171
0.9	0.051271	0.051271	0.051271
1.0	0.000000	0.000000	0.000000
Erro	< 4.3E-06	< 3.1E-06	

Tabela 5. Comparação dos valores da temperatura em vários pontos do domínio espacial, em t=0.5

De modo a verificar a precisão da solução numérica calculamos a norma máxima do vector erro. Verifica-se que a solução gerada com o código Matlab tem uma boa precisão estando concordante com o erro introduzido na posição inicial da fronteira móvel e compara favoravelmente com a solução apresentada em [8]. Por último, na tabela 6 comparam-se algumas estatísticas computacionais apresentadas pelo integrador *ode15s* com as obtidas da simulação com o código FORTRAN.

	n.passos	n.f	n.jac	n.LU	n.Sist	t.comp
LSODIS	94	871	40	40		0.18
ode15s	346	811	22	142	570	7.66

Tabela 6. Comparação de algumas estatísticas computacionais dos integradores

Observa-se que o número de passos usados neste exemplo pelo integrador *ode15s*, bem como o número de decomposições LU efectuadas são bastante maiores que (superiores a 3.5 vezes) os realizados pela LSODIS. É importante referir que 65% dos passos dados pela LSODIS ocorreram desde o início da simulação até t=0.1, contra apenas 38% no *ode15s*. Podemos concluir que, pelo menos neste exemplo, o integrador do Matlab usa passos bem mais apertados. Por outro lado, em relação ao número de avaliações de $\boldsymbol{f}$ e da sua matriz jacobiana feitas pelo *ode15s*, constata-se que é sensivelmente menor e cerca de metade, respectivamente, dos correspondentes obtidos com a LSODIS. Note-se que a LSODIS não avalia só a função $\boldsymbol{f}$ e a matriz $\partial \boldsymbol{f} / \partial \boldsymbol{Y}_{\boldsymbol{v}}$, cada avaliação envolve o cálculo do resíduo $\boldsymbol{R} = \boldsymbol{f} - \boldsymbol{M} \cdot d\boldsymbol{Y}_{\boldsymbol{v}} / dt$ e da sua matriz jacobiana. Durante a integração do sistema de EDOs o *ode15s* resolveu 570 sistemas de equações lineares algébricas. Na última coluna da tabela apresentam-se os tempos de computação em segundos das duas simulações.

6. CONCLUSÕES

Neste trabalho apresentamos uma implementação Matlab do MEFM com funções de base não lineares, para a resolução de problemas de convecção-reacção-difusão. A maior vantagem desta implementação é a simplicidade com que pode ser utilizada para resolver de modo eficiente uma vasta classe de problemas evolutivos. Os exemplos numéricos testados permitem avaliar a robustez e performance deste código numérico escrito em Matlab e baseado no MEFM. Em particular, os resultados obtidos no 1º e 3º problemas estão de acordo com a solução analítica do respectivo modelo, permitindo-nos concluir que a opção por qualquer um dos dois integradores não é determinante para a qualidade da solução numérica.

REFERÊNCIAS

[1] K. Miller and R.N. Miller, "Moving finite elements I", *SIAM J. Numer. Anal.*, Vol. **18,** No. **6**, pp. 1019-1032, (1981).

[2] K. Miller, "Moving finite elements II", *SIAM J. Numer. Anal.*, Vol. **18,** No. **6**, pp. 1033-1057, (1981).

[3] C.A. Sereno, A.E. Rodrigues and J. Villadsen, "Solution of partial differential equations systems by the moving finite element method", *Computers Chem. Engng.*, Vol. **16,** No. **6**, pp. 583-592, (1992).

[4] M.C. Coimbra, C.A. Sereno and A.E. Rodrigues, "Moving finite element method: applications to science and engineering problems", *Computers and Chem. Engng.*, **28,** pp. 597-603, (2004).

[5] J. Alberty, C. Carstensen, S.A. Funken and R. Klose, "Matlab implementation of the

finite Element method in elasticity", *Computing*, **69,** pp. 239-263, (2002).

[6] P. Saucez, L. Some and A.V. Wouwer, "Matlab implementation of a moving grid method based on the equidistribution principle", *Appl. Math. and Comput.*, **215,** pp. 1821-1829, (2009).

[7] L.F. Shampine and M.W. Reichelt, "The Matlab ODE Suite"*, SIAM J. Sci. Comput.*, **18**, pp. 1-22, (1997).

[8] R.J. Robalo, M.C. Coimbra, and A.E. Rodrigues, "Modeling time-dependent partial moving boundaries by the moving finite element method", *III European Conference on Computational Mechanics Solids, Structures and Coupled Problems in Engineering*, C.A. Mota Soares et.al. (eds.)**,** Springer, Lisboa (2006), pp. 47.

[9] W. Huang and R.D. Russel, *Adaptive moving mesh methods*, Applied Mathematical Sciences, Vol. 174**,** Springer, (2010).

[10] J.G. Verwer, J.G. Blom and J.M. Sanz-Serna, "An adaptive moving grid method for one-dimensional systems of partial differential equations", *Journal of Computational Physics*, **82,** pp. 454-486, (1989).

[11] X. Xu, "Moving mesh methods for moving boundary problems and higher order partial differential equations", Ph.D. Thesis, Simon Fraser University, Canada, (2008).