Daniel Ribeiro Favoreto

Método das diferenças finitas aplicado a problemas bidimensionais

Vitória, Espírito Santo
2016

Daniel Ribeiro Favoreto

Método das diferenças finitas aplicado a problemas bidimensionais

Relatório apresentado como requisito parcial para a obtenção de aprovação na disciplina Algoritmos Numéricos II, no curso de Ciência da Computação, na Universidade Federal do Espírito Santo.

UFES
Departamento de Informática

Vitória
Espírito Santo
2016

Resumo

O trabalho a seguir analisa a aplicação do método das diferenças finitas em problemas bidimensionais. A variação da forma de armazenamento das estruturas resultantes pela discretização da equação de transporte por diferenças finitas é analisada de forma a observar o impacto no tempo de processamento. O método foi implementado na linguagem de programação C. Experimentos computacionais são conduzidos.

Palavras-chave: Método das diferenças finitas. Problemas bidimensionais. C.

Sumário

Introdução

Este trabalho apresenta problemas físicos modelados por equações diferenciais parciais, os quais são resolvidos com o uso do método das diferenças finitas. O objetivo deste trabalho é verificar o impacto no tempo de processamento de acordo com a variação na forma de armazenamento usando aproximações de 1^a e 2^a ordem e utilizando o método SOR para resolução dos sistemas lineares.

Este trabalho é dividido em 4 partes. Na primeira parte, é discutido brevemente as técnicas e ordens de aproximação do método das diferenças finitas. A segunda parte refere-se a parte da implementação do código onde a estrutura e partes significativas comentadas são apresentadas. A terceira parte mostra os experimentos numéricos feitos, e por fim, a quarta parte apresenta as conclusões obtidas dos testes realizados na terceira parte.

Método das diferenças finitas

As aproximações de derivadas por diferenças finitas são uns dos mais simples e antigos métodos de se resolver equações diferenciais. Consistem em aproximar o operador diferencial ao substituir as derivadas na equação pelo coeficiente diferencial. O domínio é particionado no espaço e aproximações da solução são calculadas nos pontos do espaço.

Para a resolução dos problemas bidimensionais, primeiramente discretiza-se o espaço, dividindo-o em *n* pontos espaçados igualmente no eixo *x* e *m* pontos espaçados igualmente no eixo *y*. Assim, considerando o domínio = (*a, b*) × (*c, d*), os espaçamentos dos pontos do espaço discretizado serão:

$$h_x = \frac{b-a}{n-1} \tag{1}$$

$$h_y = \frac{d-c}{m-1} \tag{2}$$

Onde (1) é o espaçamento em *x* e (2) é o espaçamento em *y*. Logo, as aproximações das derivadas são dadas pelas seguintes equações:

$$\frac{\partial u}{\partial x} = \frac{u_{i+1,j} - u_{i,j}}{h_x} \tag{3}$$

$$\frac{\partial u}{\partial x} = \frac{u_{i,j} - u_{i-1,j}}{h_x} \tag{4}$$

$$\frac{\partial u}{\partial x} = \frac{u_{i+1,j} - u_{i-1,j}}{2h_x} \tag{5}$$

$$\frac{\partial u}{\partial y} = \frac{u_{i,j+1} - u_{i,j}}{h_y} \tag{6}$$

$$\frac{\partial u}{\partial y} = \frac{u_{i,j} - u_{i,j-1}}{h_y} \tag{7}$$

$$\frac{\partial u}{\partial y} = \frac{u_{i,j+1} - u_{i,j-1}}{2h_y} \tag{8}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2} \tag{9}$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_y^2} \tag{10}$$

Onde (3), (4) e (5) são, respectivamente, as diferenças finitas adiantada, atrasada e central em *x*, (6), (7) e (8) são, respectivamente, as diferenças finitas adiantada, atrasada e central em y, sendo (9) e (10)

são, respectivamente, as diferenças finitas centrais de segunda ordem em *x* e *y*.

Após substituir as diferenças finitas na equação diferencial parcial, é gerado um sistema linear *Au* = *f*, onde cada elemento ui do vetor solução é o valor aproximado da função *u* em cada ponto do espaço discretizado.

A matriz *A* é montada com os coeficientes encontrados após a aplicação das diferenças finitas na equação diferencial parcial, tendo como característica principal, no caso bidimensional, ser uma matriz pentadiagonal.

Implementação

O código deste trabalho foi implementado usando a linguagem de programação C. No arquivo main.c contém 3 funções para validação 1, validação 2 e aplicação 1. As 3 funções tem como retorno o vetor solução *u* e imprimem o tempo *t* gasto na resolução do sistema *Au* = *f*. A função main.c também gera um arquivo em formato .m para eventual plotação do gráfico.

Para a aplicação 1, para a derivada primeira da equação, foram usadas aproximações de 2ª ordem (central em *x* e central em *y*) e aproximações de 1ª ordem (adiantada em *x* e atrasada em *y*) para a montagem da matriz A do sistema.

$$-k\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+\beta_x(x,y)\frac{\partial u}{\partial x}+\beta_y(x,y)\frac{\partial u}{\partial y}+\gamma(x,y)u=f(x,y) \quad \text{em } \Omega \quad (1)$$

Na validação 2, a expressão (1) foi particionada em somas para facilitar a modelagem do problema. Portanto, temos as seguintes equivalências:

$$-k\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) = -e^{x^{4.5}}\left(90.x^{4.5}(y-1.)y+247.5(x-1.)x^{3.5}(y-1.)y+202.5(x-1.)x^{8.}(y-1.)y+20(x-1)x+20(y-1)y\right)$$

$$\beta_x(x,y)\frac{\partial u}{\partial x} = -45\,e^{x^{4.5}}\left(-x^{5.5}+x^{4.5}-0.444444\,x+0.222222\right)(y-1)y$$

$$\beta_y(x,y)\frac{\partial u}{\partial y} = 20\,y\,10\,e^{x^{4.5}}(x-1)x(2y-1)$$

$$\gamma(x,y)u = 10xy(1-x)(1-y)e^{x^{4.5}}$$

$$f(x,y) = 10xy(1-x)(1-y)e^{x^{4.5}}$$

Na validação 2, o programa faz o preenchimento dos valores prescritos, caso haja. Por motivos de simplificação, a seguir é mostrado o trecho de preenchimento da linha direita ou limite direita.

```
// linha direita
   if (valor_prescrito_na_linha_da_direita) {

    for (i = 1; i < m - 1; i++) {

        index = i*n + (n-1);

        X.v[index] = valorPrescritoDireita;
        b.v[index] = valorPrescritoDireita;

    }
}
```

Os cálculos de a1,b1,c1 foram realizados fora da malha, uma vez que nesse teste o Betax é constante e os cálculos de d1 e e1 foram realizados dentro da iteração sobre a malha, já que Betay = 20*y.

```
double a1, b1, c1, d1, e1;

// validacao 2
a1 = 1.0 + (2*k * (1.0/(hx*hx) + (1.0/(hy*hy))));
// e1 sendo calculado dentro do for
c1 = (-k/(hx*hx)) - (1.0/(2*hx));
b1 = (-k/(hx*hx)) + (1.0/(2*hx));
d1 = 0.0;
e1 = 0.0;
// d1 sendo calculado dentro do for
```

Foi utilizado um laço de repetição do while() iterando no máximo ITERATIONMAX e enquanto a flag de convergência não for setada indicando que o sistema já convergiu.

Dentro desse laço do while() é feito os tratamentos de contorno da malha especialmente nos cantos superior direito, superior esquerdo, inferior esquerdo e inferior direito. Assim como os 4 limites da malha foram tratados, limite esquerdo, limite direito, limite acima e limite abaixo.

A seguir tem-se o tratamento do canto inferior direito:

```
// canto inferior direito
if (!valor_prescrito_na_linha_da_direita && !valor_prescrito_na_linha_de_baixo) {

    index = n-1;

    e1 = kSobreHyQuadrado;
    d1 = kSobreHyQuadrado;

    // validacao 1 e validacao 2
    novoA1 = a1 + b1 + e1;

    valorAtualizado = (e1*(-derivadaNormalY)*hy) + b1*(derivadaNormalX)*hx;

    X.v[index] = X.v[index] + omega * ( ((b.v[index] - valorAtualizado) - (c1*X.v[index - 1] + d1*X.v[index + n]))/novoA1
dex]);

}
```

O tratamento do limite esquerdo foi realizado da seguinte maneira:

```
  for (i = 1; i < m - 1; i++){

        double parcela = (20*(i*hy)/(2*hy));

        d1 = kSobreHyQuadrado + parcela;
        e1 = kSobreHyQuadrado - parcela;

      // linha esquerda
      if (!valor_prescrito_na_linha_da_esquerda){

          index = i*n;


          novoA1 = a1 + c1;
          valorAtualizado = c1*(-derivadaNormalX)*hx;

          X.v[index] = X.v[index] + omega * (((b.v[index] - valorAtualizado) - (e1*X.v[index - n] + b1*X.v[index + 1] + d1*X.v[index
+ n]))/novoA1 - X.v[index]);

      }
```

Já o cálculo da maior diferença e maior valor foram calculados da seguinte maneira:

```
for (i = 0; i < m*n; i++){

    diferencaAtual = fabs(X.v[i] - xAntigo.v[i]);

    if (diferencaAtual > maxDiferenca){

        maxDiferenca = diferencaAtual;

    }

    if (fabs(X.v[i]) > maxValor){

        maxValor = fabs(X.v[i]);

    }

  xAntigo.v[i] = X.v[i];

}
```

Em seguida o cálculo do erro aproximado:

```
erro = maxDiferenca/maxValor;

if (erro > alpha)
    nao_convergir = 1;
else
    nao_convergir = 0;

maxDiferenca = -1;
maxValor = -1;
```

E por final tanto as funções de validação 1 e validação 2 quanto da aplicação 1 retornam o vetor solução.

Experimentos Numéricos

Experimentos numéricos foram conduzidos com o uso da tabela fornecida nas especificações deste trabalho. Para cada discretização, foi calculado o tempo necessário para a resolução do sistema gerado. A tabela 1 mostra as instâncias de teste escolhidas para a realização dos experimentos.

A seguir tem-se a tabela de testes para a validação 2, utilizando-se m e n variados para cada tamanho do sistema linear. No seguinte teste foi utilizado os seguintes valores de ALPHA = 0.00001, OMEGA = 1.6, K = 1.0, NÚMERO DE ITERAÇÕES MÁXIMAS = 10000 e VALORES PRESCRITOS = 0.

n	m	Ordem do sistema linear	Tempo (s)	Iterações
50	50	2500	0.057320	313
25	100	2500	0.067824	603
250	40	10000	0.279137	3008
100	100	10000	0.315509	1102
1000	500	500000	137.954501	10000
1000	1000	1000000	365.347596	10000
2500	4000	10000000	6900.132692	10000

Tabela 1

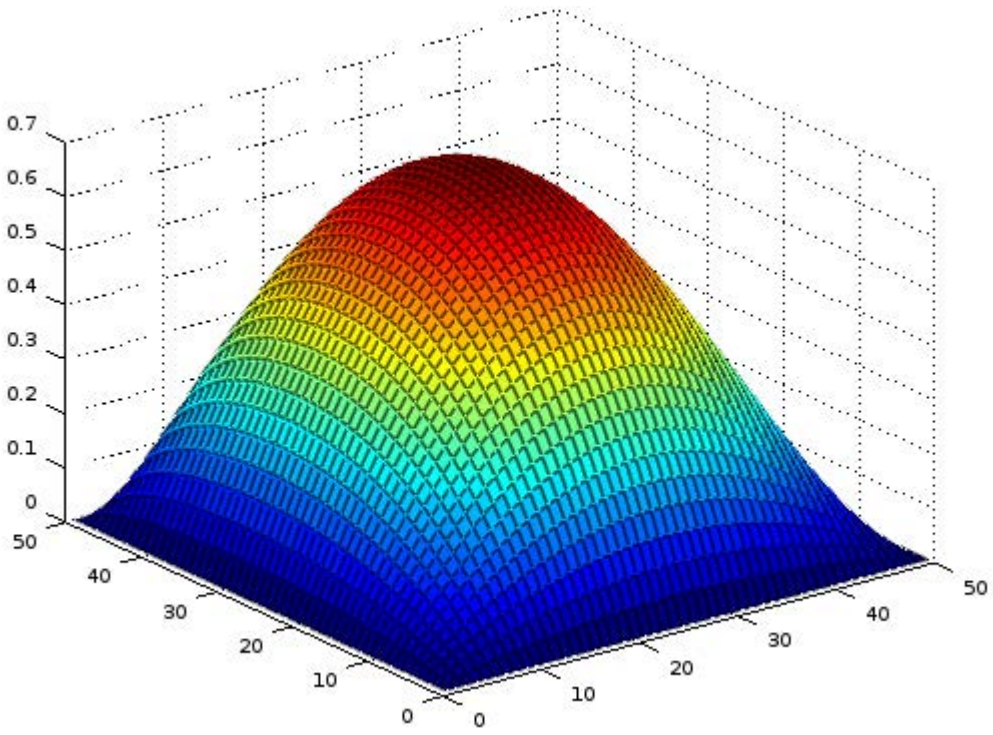

Gráfico para o resultado na validação 2 tamanho: 50x50

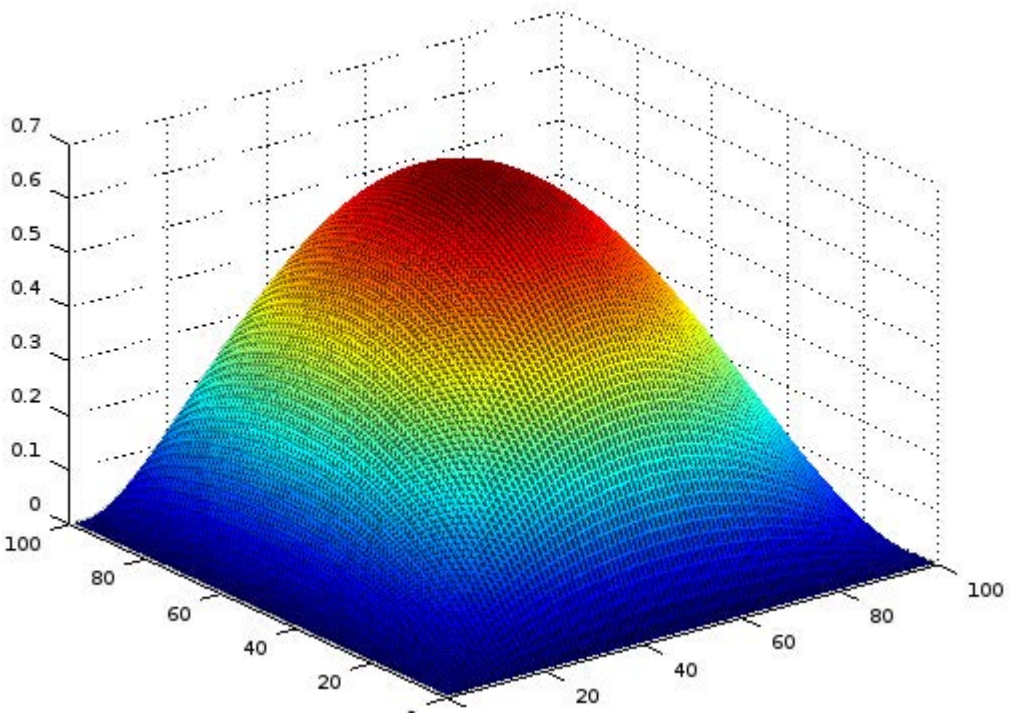

Gráfico para validação 2 tamanho: 100x100

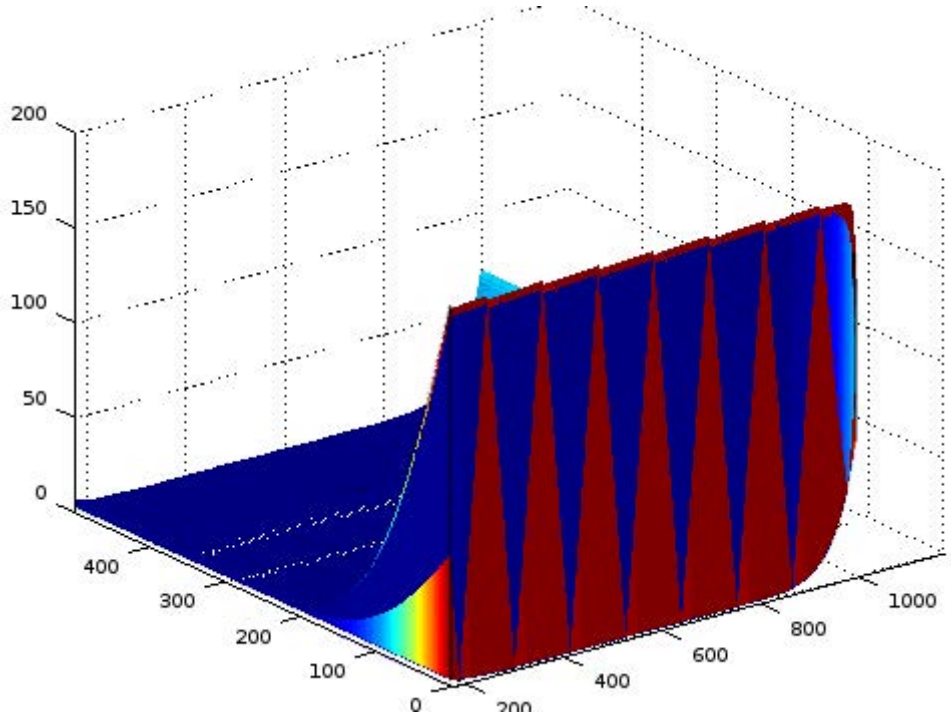

Gráfico para aplicação 1 tamanho: 1000x500

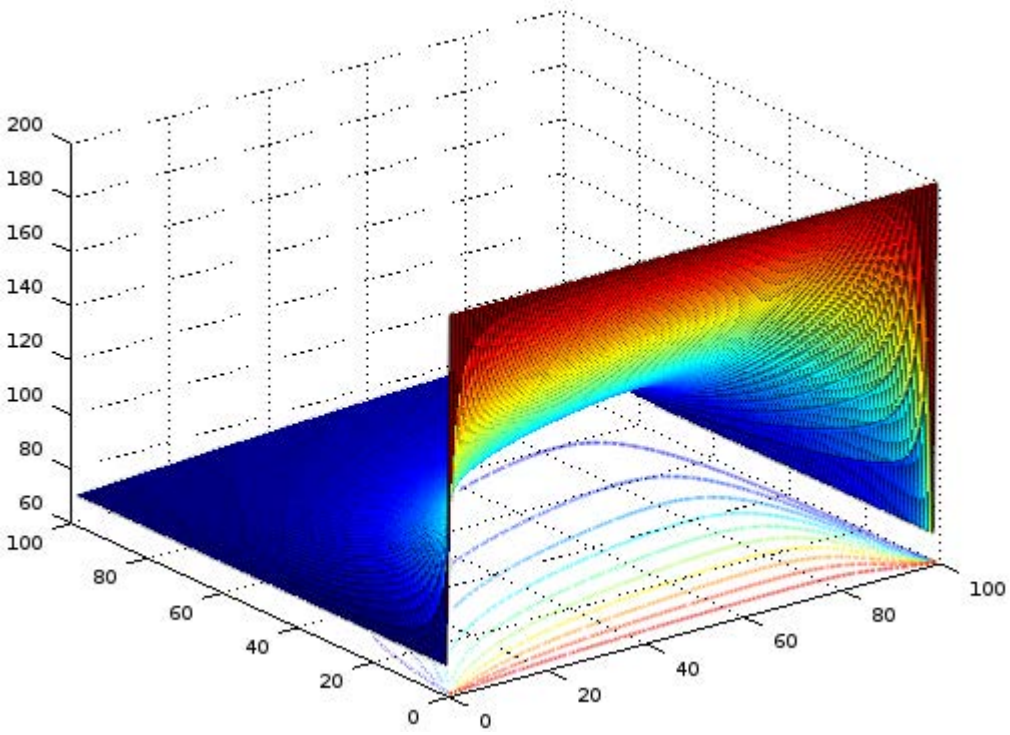

Gráfico para aplicação 1 tamanho: 100x100

Conclusão

Pouco se pode concluir a partir do que foi feito, uma vez que não houve tempo hábil para realizar todas as aplicações de forma satisfatória assim como os testes computacionais.

Porém, observou-se que a forma de armazenamento das matrizes impactava principalmente no tempo de execução dos problemas de validação e da aplicação 1.

O aluno também pôde observar que alguns gráficos, apesar de não terem sido publicados no relatório, apresentaram diferenças quando compara-se instâncias menores com instâncias maiores, devido ao espaçamento hx e hy serem maiores quando se tem tamanhos menores de entrada, gerando instabilidade no sistema. Conforme maiores os subintervalos m e n, os valores de hx e hy diminuem tornando o sistema mais estável.

Obviamente quanto maior fosse a quantidade de iterações máxima, melhor o sistema convergia embora o desempenho sofresse um grande impacto.

Em cada teste foi utilizado uma máquina Core i5,2.5Ghz e 4Gb RAM em um sistema operacional Ubuntu 16.04.

Referências

CAUSON, D. M., MINGHAM C. G., *Introductory Finite Di erence Methods for PDEs*, 2010. Disponível em: hhttp://www.leka.lt/sites/default/files/dokumentai/introductory-finite-di
erence-methods-for-pdes.pdfi. Acesso em 18.04.2016.

EBERLY, D. *Derivative Approximations by Finite Di erences*, 2001. Disponível em: hhttp://www.geometrictools.com/Documentation/FiniteDi
erences.pdfi. Acesso em 18.04.2016.