Ministério da
Ciência, Tecnologia e Inovação

sid.inpe.br/mtc-m18@80/2009/10.21.23.39-RPQ

MINERAÇÃO DOS DADOS DE ENTRADA DE UMA ESTRUTURA DE DADOS BASEADA EM ÁRVORE QUARTENÁRIA PARA A RESOLUÇÃO DE EDP'S UTILIZANDO WAVELET

Marilyn Menecucci Ibañez

Relatório Final da Disciplina Princípios e Aplicações de Mineração de Dados (CAP-359) do Programa de Pós-Graduação em Computação Aplicada, ministrada pelo Professor Rafael Santos

URL do documento original:
<http://urlib.net/8JMKD3MGP8W/369N6FB>

INPE
São José dos Campos
2012

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br

Ministério da
Ciência, Tecnologia e Inovação

sid.inpe.br/mtc-m18@80/2009/10.21.23.39-RPQ

MINERAÇÃO DOS DADOS DE ENTRADA DE UMA ESTRUTURA DE DADOS BASEADA EM ÁRVORE QUARTENÁRIA PARA A RESOLUÇÃO DE EDP'S UTILIZANDO WAVELET

Marilyn Menecucci Ibañez

Relatório Final da Disciplina Princípios e Aplicações de Mineração de Dados (CAP-359) do Programa de Pós-Graduação em Computação Aplicada, ministrada pelo Professor Rafael Santos

URL do documento original:
<http://urlib.net/8JMKD3MGP8W/369N6FB>

INPE
São José dos Campos
2012

SUMÁRIO

LISTA DE FIGURAS

Pág.

CAPÍTULO 1

INTRODUÇÃO

A utilização das técnicas wavelets em diversas áreas da ciência tem ganhado muita importância principalmente na área de análise numérica. A solução de EDP's por meio das técnicas wavelets é de grande interesse para o desenvolvimento de métodos adaptativos.

Os métodos adaptativos apresentam as soluções como refinamento da sua entrada de dados. Este refinamento depende da regularidade em um local específico do dado.

As estruturas de dados, tal como as árvores, estão sendo bastante aplicadas na realização do refinamento dos dados de uma análise numérica. Para isto vários modelos de estruturas tem sido utilizados, como: *binary tree*, *quadtree*, *octtree*, etc. Neste projeto utiliza-se a estrutura de dados *quadtree* na implementação do refinamento. Uma *quadtree* é, basicamente, uma estrutura de árvore que possui quatro filhos.

A mineração de dados é uma área que facilita o estudo e extração de informação de uma grande base de dados. Com base nesse contexto, o objetivo deste projeto é utilizar a mineração de dados para analisar e relacionar os dados de entrada para o refinamento e análise de imagens por meio de malhas adaptativas. O objetivo da análise dos dados de entrada é encontrar uma relação que possa de alguma forma ser utilizada para uma avaliação destes dados. No projeto utiliza-se os métodos de árvore de decisão e redes neurais para realizar a avaliação dos dados.

Para o desenvolvimento do projeto, utiliza-se a orientação a objetos aplicada a linguagem de programação C++, a biblioteca Blitz++, que facilita a manipulação de *arrays*, na IDE Kdevelop e o software de mineração de dados *Weka*. Para um melhor entendimento do projeto, este relatório está dividido da seguinte forma: No Capítulo 2 é apresentado a metodologia, os dados e as ferramentas utilizados no projeto, os resultados obtidos são apresentados no 3 e por fim, a conclusão.

CAPÍTULO 2

METODOLOGIA E DADOS

A técnica aplicada no desenvolvimento do projeto consiste na entrada de uma função que gere um *array* e seu refinamento. O processo de refinamento só é realizado dependendo da regularidade da malha em um determinado local. No programa em liguagem C++ desenvolvido para simular esta técnica, utiliza-se a estrututra de dados *quadtree* e inicialmente cria-se o nó principal (*root*) da árvore que recebe a malha (NxN). Para a aplicação do refinamento realiza-se uma subdivisão temporária da malha em quatro novos blocos. Nestes blocos são armazenados as malhas referentes aos respectivos quadrantes da malha principal. No preenchimento das malhas utiliza-se somente as posições ímpares da matriz. Para mineração de dados são gerados arquivos em formas de tabelas e no formato *.arff* que é um padrão aceito pelo software *Weka*.

2.1 Dado de Entrada

A entrada de dados do programa é feita por meio de uma equação que representa uma malha, a ordem da matriz a ser analisada, os parâmetros da equação, tais como alpha (parâmetro que compõe a equação de entrada), quantidade total níveis da árvore gerada, quantidade de nós refinados, épsilon (valor utilizado para a verficação da necessidade do refinamento da malha).

2.2 Equação

No projeto utiliza-se dois tipos de equações, a equação (2.1) que representa uma *Gaussiana* e a equação (2.2) que representa uma função tipo pulso.

$$f(x,y) = \alpha e^{(100*((x-0.5)^2+(y-0.5)^2))} \tag{2.1}$$

$$f(x,y) = \alpha e^{(-2500*((x-0.3)^2+(y-0.3)^2))+\sin(2\pi x)+\sin(2\pi y)} \tag{2.2}$$

2.3 Malha Adaptativa

A Malha Adaptativa é uma técnica que representa a solução por meio de refinamento de uma entrada de dados [6]. O processo de refinamento é realizado em determinado local da malha dependendo da regularidade que este apresenta. A Figura 2.1 mostra um exemplo de uma malha adaptativa.

0.135335	0.209611	0.286505	0.345591	0.367879
0.209611	0.324652	0.443747	0.535261	0.569783
0.286505	0.443747	0.606531	0.731616	0.778801
0.345591	0.535261	0.731616	0.882497	0.939413
0.367879	0.569783	0.778801	0.939413	1

FIGURA 2.1 – Matriz gerada pela função 2.1.

2.4 Estrutura de Dados *QuadTree*

A árvore quaternária é uma estrutura de dados em que cada nó possui no máximo quatro filhos. Um nó é dito nó folha quando não possui filhos. As árvores são definidas como completas quando o seu último nível possui o número máximo de elementos; caso contrário são definidas como incompletas. A árvore quaternária pode ser utilizada em diversas áreas, principalmente na área gráfica, como exemplo em processamento digital de imagens, computação gráfica, banco de dados de imagens entre outras. A Figura 2.2 apresenta um exemplo de uma árvore quaternária completa, em que p é o nó raiz e $4p + 1, 4p + 2, 4p + 3$ e $4p + 4$ seus filhos.

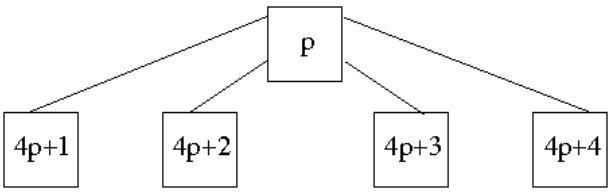

FIGURA 2.2 – Exemplo de uma árvore quaternária completa em que p é o nó raiz e $4p+1, 4p+2, 4p+3$ e $4p+4$ seus filhos.

A Figura 2.3 apresenta um exemplo da representação de uma malha adaptativa em uma árvore quaternária . Nesta figura também é mostrado com é feita a determinação da posição (x, y) de cada nó na malha. A posição do nó raiz é utilizada como base para o início do cálculo das posições dos novos nós, e é determinada pela posição central, $(x_1; x_2) = (0, 5; 0, 5)$. No esquema a seguir tem-se: (a) determinação da posição $(x_1; x_2)$ dos nós em uma malha, (a) e (b) bloco de malhas refinada, na qual o seu máximo refinamento encontra-se no segundo nível do segundo quadrante com sua respectiva árvore quaternária incompleta.

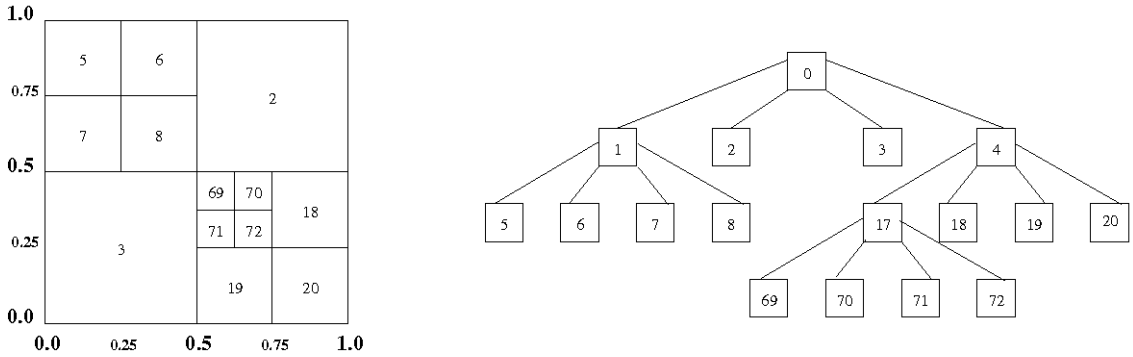

FIGURA 2.3 – No esquema a seguir tem-se: (a) determinação da posição $(x_1; x_2)$ dos nós em uma malha, (a) e (b) bloco de malhas refinadas, na qual o seu máximo refinamento encontra-se no segundo nível do segundo quadrante com sua respectiva árvore quaternária incompleta.

2.4.1 Interpolação dos Dados

A interpolação dos dados pode ser feita de forma linear ou cúbica. As novas malhas são preenchidas por meio da interpolação linear dos pontos das posições ímpares da matriz. Este tipo de interpolação é realizado pelo cálculo da média aritmética dos valores utilizados neste processo. Cada valor a ser interpolado é comparado com uma constante épsilon, se um valor interpolado de um quadrante for maior, por exemplo, que a constante então um nó é alocado, uma nova malha (NxN) gerada e a malha analisada é refinada na naquele quadrante. O processo prossegue até não se satisfazer a condição da constante ou se alcançar um nível máximo estipulado de refinamento. A seguir é apresentado um exemplo da interpolação de um quadrante da malha da Figura 2.1. A Figura 2.4 apresenta a divisão em quatro quadrantes da malha. O quandrante utilizado para a ilustração do processo é o posicionado no topo a esquerda. A Figura 2.5 mostra a disposição dos valores do primeiro quadrante da malha mãe nas posições ímpares da nova malha a ser gerada. E por fim, a Figura 2.6 apresenta a nova malha com os valores das posições pares interpolados. Observa-se nesta figura que os valores entre posições pares são calculados utilizando os valores interpolados dessas posições pares.

2.4.2 Arquivo dos Dados de Saída

A seguir são apresentados a estrutura dos arquivos de saída de dados. Os arquivos são gerados com duas estruturas distintas, uma em forma de tabela, Anexo A, e outro no formato .arff, Anexo B, aceito pelo software *Weka*

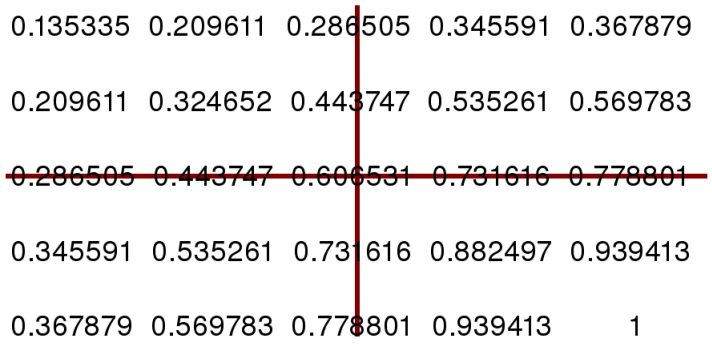

FIGURA 2.4 – Divisão da malha em quatro quadrantes.

0.135335	0.209611	0.286505
0.209611	0.324652	0.443747
0.286505	0.443747	0.606531

FIGURA 2.5 – Disposição dos valores do primeiro quadrante nas posições ímpares da nova malha.

0.135335	0.172	0.209611	0.248	0.286505
0.172	0.219	0.267	0.316	0.365
0.209611	0.267	0.324652	0.384	0.443747
0.248	0.316	0.384	0.454	0.525
0.286505	0.365	0.443747	0.525	0.606531

FIGURA 2.6 – Interpolação linear dos valores das posições ímpares.

2.4.2.1 Árvore de Decisão

As árvores de decisão são um conjunto de testes realizados sobre os atributos de uma base de dados que determinam a classe de um dado a partir das respostas fornecidas às questões. Nos testes são considerados os nós da árvore e as classes são suas folhas. Cada teste pode resultar em uma folha (decisão do classificador) ou em um novo nó(novo teste em outro atributo) [8].

2.4.2.2 Redes Neurais

Os modelos neurais, procuram aproximar o processamento dos computadores ao cérebro. As redes neurais possuem um grau de interconexão similar a estrutura do cérebro e um computador convencional moderno a informação é transferida em tempos específicos dentro de um relacionamento com um sinal para sincronização [2].

2.5 Ferramentas Utilizadas

2.5.1 Orientação a Objetos

O termo orientação a objetos pressupõe uma organização de software em termos de coleções de objetos discretos incorporando estrutura e comportamento próprios. Essa abordagem modela o mundo real com classes e instâncias. Cada classe é a estrutura de uma variável, ou seja, um tipo de dado. Nela, são declarados atributos e métodos que poderão ser executados ou acessados nas instâncias da mesma classe. As classes possuem uma função muito importante na modelagem orientada a objetos, elas dividem o problema, modularizam a aplicação e baixam o nível de acoplamento do software. Abaixo são apresentadas uma série de definições sobre a arquitetura de orientação a objetos:

- **Instância, objeto:** Uma variável do tipo de uma classe.
- **Construtor:** Responsável por iniciar a criação e inicialização de uma instância de classe.
- **Método:** Funções referenciadas àquela classe.
- **Hierarquia de classes:** Um grupo de classes que estão relacionadas por herança.
- **Superclasse:** Classe que é estendida por uma determinada classe.
- **Subclasse:** Classe que estende determinada classe.
- **Classe base:** Classe de determinada hierarquia que é uma superclasse de todas as outras classes. A classe pai de todas as classes.

2.5.2 IDE Kdevelop

No desenvolvimento do programa utilizou-se a linguagem C++ [9] no ambiente de desenvolvimento Kdevelop 3.5.3 para a distribuição Ubuntu 9.04 do sistema operacional Linux. Esta ferramenta possibilita uma maior organização das classes e arquivos do programa, praticidade para controlar as versões do programa por meio do controlador de versões Subversion e facilidade na depuração e compilação do código.

2.5.3 Blitz++

Blitz++ é uma biblioteca da linguagem C++ para ciência da computação. Atualmente, fornece funções para o trabalho com *arrays* e vetores, número randômicos, etc [1].

2.5.4 Mineração de Dados - (Data-Mining)

Data Mining ou Mineração de Dados consiste em um processo analítico projetado para explorar grandes quantidades de dados na busca de padrões consistentes e/ou relacionamentos sistemáticos entre variáveis e, então, validá-los aplicando os padrções detectados a novos subconjuntos de dados. O processo consiste basicamente em 3 etapas: exploração, construção de modelo ou definição do padrão, e validação/verificação [5]. A Figura 2.7 apresenta as etapas da mineração de dados.

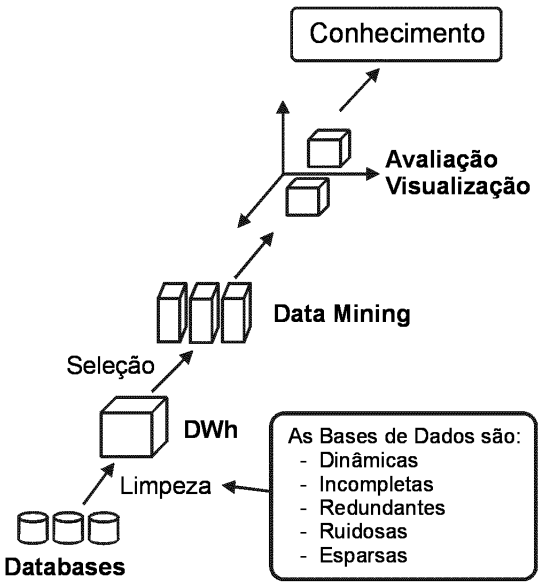

FIGURA 2.7 – Etapas da Mineração de Dados.

FONTE: [7].

2.5.5 Software Weka

O Weka realiza a análise computacional e estatística dos dados fornecidos recorrendo as técnicas de data-mining tentando, indutivamente, a partir dos padrões encontrados, gerar hipóteses para soluções e inclusive teorias sobre os dados em questão [3]. A Figura 2.8 apresenta a janela de ínicio do programa.

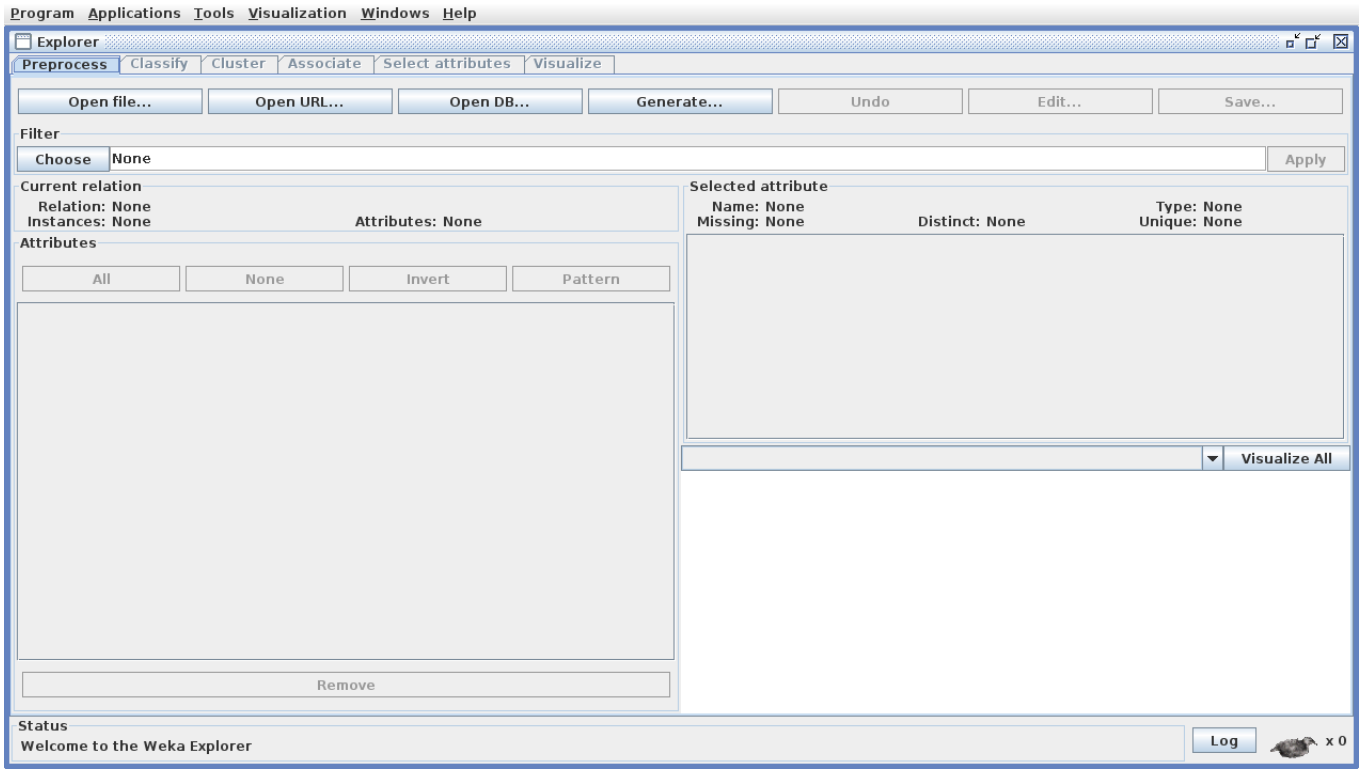

FIGURA 2.8 – Janela inicial do programa *Weka*.

CAPÍTULO 3

RESULTADOS E ANÁLISES

Os resultados obtidos compreendem o desemvolvimento de um programa em linguagem C++, a criação da árvore de decisão, uma rede neural e de gráficos que auxiliam na interpretação dos dados e que representam a análise dos arquivos de saída dos dados pelo software *Weka*.

3.1 Desenvolvimento do Programa

O programa foi desenvolvido de acordo com a modelagem elaborada e utilizando a linguagem de programação C++ na IDE Kdevelop 3.5. As classes desenvolvidas no programa até o presente momento foram wNode(), wTree(), wInterwavelet(), wNeighbor(). A seguir é apresentada uma explicação sobre como desenvolveu-se cada classe.

- **Classe wNode():** Esta classe contém a estrutura do nó da árvore. Cada nó possui informações sobre o número do índice, a matriz de valores, posição *xyz* do nó na árvore e ponteiro para o próximo bloco.

- **Classe wTree():** A classe wTree() contém as funções de criação e manipulação da árvore. A classe wTree() herda as características da classe wNode(). Nesta classe utiliza-se a técnica de programação dinâmica para auxiliar na construção da árvore por nível. Esta técnica permite por meio de uma tabela armazenar dados para serem usados posteriormente. Neste caso, são armazenado os endereços de memória de todos os nós criados em um nível. Estes dados de memória são utilizados para determinar os próximos nós a serem criados (filhos dos nós da tabela). A cada loop os nós da tabela são substituídos pelos seus filhos. As principais funções presentes nesta classe são: wConvert(), wReadFile(), wresizeVect(), wresizenewVect(), winsertVector(), wgetVector(), wCreateTree().

  wConvert(): Converte um número em ponto flutuante para inteiro.

  wReadFile(): Realiza a leitura do arquivo de entrada que possui a matriz quadrada NxN.

  wresizeVect(): Redimensiona o vetor denominado *vect* de acordo com a necessisdade do programa.

  wresizenewVect(): Redimensiona o vetor auxiliar denominado *newvect* de acordo com a necessisdade do programa.

  winsertVector(): Insere os novos nós alocados no vetor *vect*.

  wgetVector(): Retorna o vetor gerado pela função winsertVector().

wCreateTree(): Gera a árvore utilizando alocação dinâmica de memória. A cada novo nó inserido na árvore o campo de informação, no qual se insere o valor da matriz, do seu nó pai é anulado. Desta forma, somente os nós folhas da árvore contém a matriz, que é preenchida fazendo-se a interpolação.

- **Classe wInterwavelet():** Nesta classe é calculada a interpolação dos pontos ímpares da matriz. Estão sendo implementados dois tipos de interpolação (linear e cúbica). Esta classe possui as seguintes funções functionInterpolation(), checkMatriz(), checkFunction().

  functionInterpolation(): Esta função calcula a interpolação dos pontos ímpares utizando pontos pares da matriz de dados. A função apresenta dois tipos de interpolação a linear, que utiliza dois pontos pares da matriz para o cálculo e a cúbica que utiliza quatro pontos pares. Para o cálculo também deve se considerar a posição dos pontos da matriz.

  checkMatriz(): Nesta função verifica-se se o valor calculado na interpolação é maior ou menor a uma constante *épsilon*. Caso a condição não seja satisfeita o valor caculado na interpolação não é inserido na matriz.

  checkFunction: Compara a função de avaliação com o valor *épsilon*

3.2 Árvore de Decisão

A seguir é apresentado a árvore de decisão criada pelo *Weka* utilizando a base de dados gerada pelo programa desenvolvido na linguragem de programação C++. Observa-se na árvore de decisão que algumas combinações de *alpha* e *épsilon* apresentam valores diferentes de 0 e 1364, que são os valores aplicados ao projeto. O *Weka* utilizou o algoritmo $J48$ para desenvolver a árvore [4].

```
 === Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     equacao
Instances:    40000
Attributes:   5
              alfa
              level
              node
              \'epsilon
              class
Test mode:    user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===

J48 pruned tree
------------------

\'epsilon <= 2.455
|   alfa <= 345
|   |   alfa <= 30
|   |   |   \'epsilon <= 2.40001
|   |   |   |   \'epsilon <= 0.10001
|   |   |   |   |   \'epsilon <= 0.00001: 2 (143.0)
|   |   |   |   |   \'epsilon > 0.00001
```

```
|   |   |   |   |   |   \'epsilon <= 0.055: 1 (158.0)
|   |   |   |   |   |   \'epsilon > 0.055: 2 (143.0)
|   |   |   |   \'epsilon > 0.10001
|   |   |   |   |   \'epsilon <= 0.155: 1 (158.0)
|   |   |   |   |   \'epsilon > 0.155
|   |   |   |   |   |   \'epsilon <= 0.30001
|   |   |   |   |   |   |   \'epsilon <= 0.20001: 2 (143.0)
|   |   |   |   |   |   |   \'epsilon > 0.20001
|   |   |   |   |   |   |   |   \'epsilon <= 0.255: 1 (158.0)
|   |   |   |   |   |   |   |   \'epsilon > 0.255: 2 (143.0)
|   |   |   |   |   |   \'epsilon > 0.30001
|   |   |   |   |   |   |   \'epsilon <= 0.355: 1 (158.0)
|   |   |   |   |   |   |   \'epsilon > 0.355
|   |   |   |   |   |   |   |   \'epsilon <= 0.50001
|   |   |   |   |   |   |   |   |   \'epsilon <= 0.40001: 2 (143.0)
|   |   |   |   |   |   |   |   |   \'epsilon > 0.40001
|   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.455: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   \'epsilon > 0.455: 2 (143.0)
|   |   |   |   |   |   |   |   \'epsilon > 0.50001
|   |   |   |   |   |   |   |   |   \'epsilon <= 0.555: 1 (158.0)
|   |   |   |   |   |   |   |   |   \'epsilon > 0.555
|   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.70001
|   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.60001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   \'epsilon > 0.60001
|   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.655: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 0.655: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   \'epsilon > 0.70001
|   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.755: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   \'epsilon > 0.755
|   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.90001
|   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.80001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 0.80001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.855: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 0.855: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 0.90001
|   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 0.955: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 0.955
|   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.10001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.00001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.00001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.055: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.055: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.10001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.155: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.155
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.30001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.20001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.20001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.255: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.255: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.30001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.355: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.355
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.50001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.40001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.40001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.455: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.455: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.50001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.555: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.555
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.70001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.60001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.60001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.655: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.655: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.70001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.755: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.755
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.90001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.80001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.80001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.855: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.855: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.90001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 1.955: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 1.955
```

```
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 2.10001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 2.00001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 2.00001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 2.055: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 2.055: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 2.10001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 2.155: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 2.155
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 2.20001: 2 (143.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 2.20001
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 2.355
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 2.255: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 2.255
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon <= 2.30001: 2 (143.0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 2.30001: 1 (158.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   \'epsilon > 2.355: 2 (143.0)
|   |   |   \'epsilon > 2.40001: 1 (158.0)
|   |   alfa > 30
|   |   |   node <= 136: 1 (23105.0/145.0)
|   |   |   node > 136
|   |   |   |   \'epsilon <= 0.005: 1 (461.0/7.0)
|   |   |   |   \'epsilon > 0.005
|   |   |   |   |   \'epsilon <= 0.055: 1 (136.0)
|   |   |   |   |   \'epsilon > 0.055: 2 (23.0)
|   alfa > 345: 2 (3750.0)
\'epsilon > 2.455: 2 (5000.0)

Number of Leaves  :  56

Size of the tree :  111


Time taken to build model: 8.99 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances        7500               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0043
Root mean squared error                  0.0054
Relative absolute error                  0.9781 %
Root relative squared error              1.1345 %
Total Number of Instances             7500
```

3.3 Rede Neural

No desenvolvimento da rede neural o programa *Weka* utilizou o algoritmo *MultLayerPerceptron*. Da mesma forma que a árvore de decisão a rede neural apresenta algumas combinações de *alpha* e *épsilon* apresentam valores diferentes de 0 e 1364 que são os únicos aplicados ao projeto.

```
 === Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation:     equacao
Instances:    40000
Attributes:   5
              alfa
              level
              node
              \'epsilon
              class
Test mode:    user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===

Sigmoid Node 0
    Inputs    Weights
    Threshold    -57.24514892281209
```

```
    Node 2    32.63971791890457
    Node 3    25.59240387093974
    Node 4    27.36699023681204
Sigmoid Node 1
    Inputs    Weights
    Threshold    57.24514892281073
    Node 2    -32.63971791890386
    Node 3    -25.59240387093907
    Node 4    -27.366990236811382
Sigmoid Node 2
    Inputs    Weights
    Threshold    -1.824528999466709
    Attrib alfa    -12.985621904450442
    Attrib level    -0.0012589494071673507
    Attrib node    6.201963692124568
    Attrib \'epsilon    -1.2689960199735306
Sigmoid Node 3
    Inputs    Weights
    Threshold    5.884455364453955
    Attrib alfa    2.900430300890624
    Attrib level    0.015060624527394956
    Attrib node    0.42649224108309763
    Attrib \'epsilon    -86.02010175240228
Sigmoid Node 4
    Inputs    Weights
    Threshold    118.42865777406755
    Attrib alfa    311.1535818793553
    Attrib level    -0.00329135406107145
    Attrib node    -99.57151246336613
    Attrib \'epsilon    1.3062562900225414
Class 1
    Input
    Node 0
Class 2
    Input
    Node 1


Time taken to build model: 58.87 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances        7240               96.5333 %
Incorrectly Classified Instances       260                3.4667 %
Kappa statistic                          0.9201
Mean absolute error                      0.0373
Root mean squared error                  0.1418
Relative absolute error                  8.5152 %
Root relative squared error             30.0491 %
Total Number of Instances             7500
```

3.4 Gráficos

Os gráficos gerados pelo *Weka* apresentam as informações combinadas uma a uma. Por meio destes gráficos foi possível perceber comportamentos dos dados que são de grande importância para a pesquisa da proponente do trabalho. Esses comportamentos se caracterizam pela falta, excesso e presença de dados e apresentam o intervalo numérico que os parâmentro *alpha* e *épsilon* devem estar para se gerar uma árvore quaternária adequada a resolução da equação dos dados analisados. As Figuras 3.1 e 3.2 apresentam a relação entre os parâmetros *alpha*, *level*, *node*, *épsilon* e *tipo/class* por meio de gráficos gerados na mineração dos dados. Nesses gráficos a cor azul representa a equação (2.1) e a vermelha a (2.2) da seção 2.2.

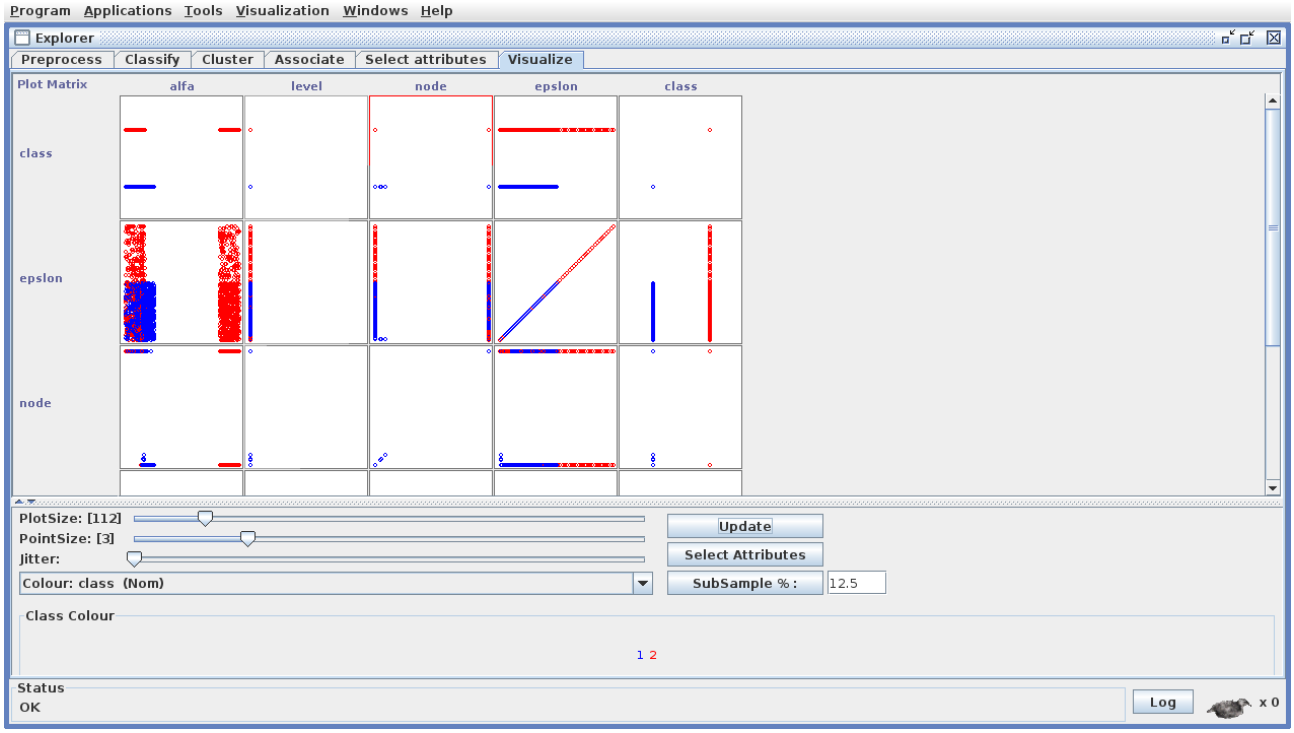

FIGURA 3.1 – Relação entre os parâmetros de entrada *node*, *épsilon* e *tipo/class* com os outros parâmetros. Cor azul: equação (2.1) e vermelha (2.2) da seção 2.2

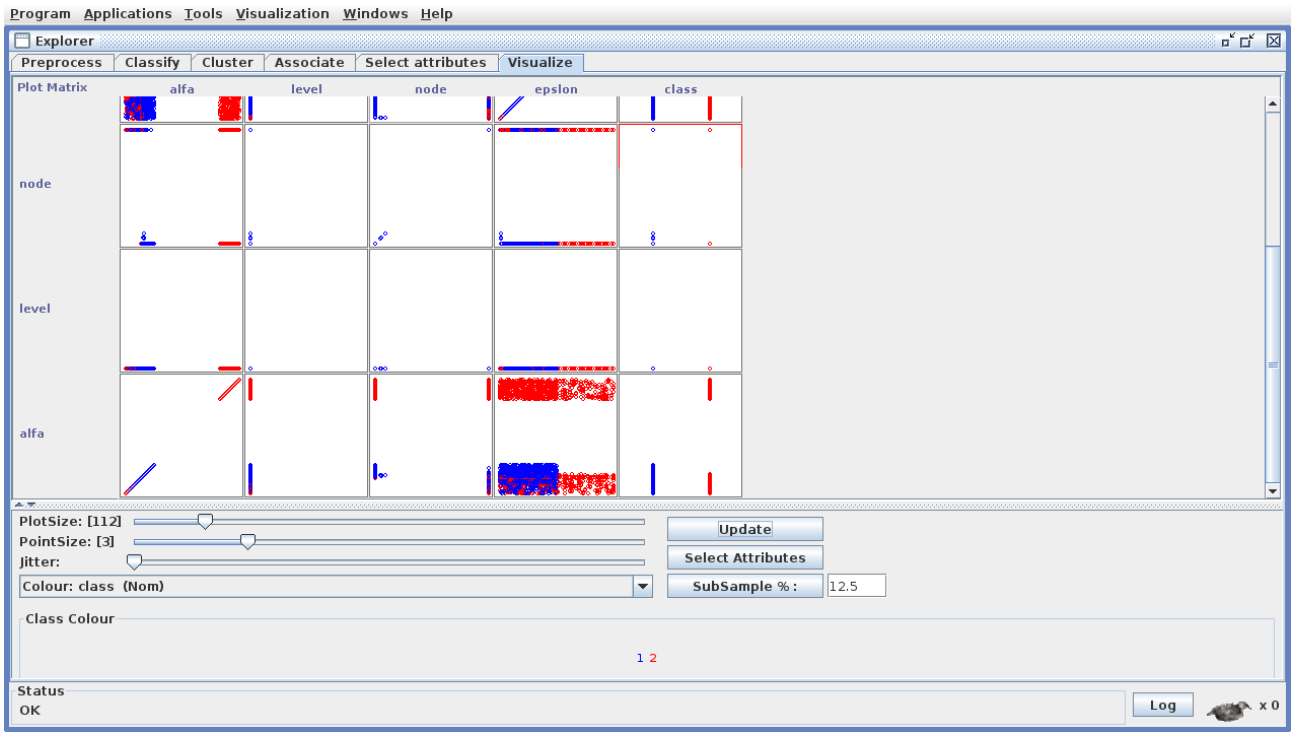

FIGURA 3.2 – Relação entre os parâmetros de entrada *alpha*, *level*, *node* com os outros parâmetros. Cor azul: equação (2.1) e vermelha (2.2) da secção 2.2

CAPÍTULO 4

CONSIDERAÇÕES FINAIS

4.1 Conclusão

No desenvolvimento do projeto obteve-se como resultados a criação da árvore de decisão, a rede neural e os gráfcos. Por esses métodos é possível observar que dependendo da combinação de *épsilon* e *alpha* a quantidade de nós refinados pode variar entre zero (nenhum nó refinado), um valor intermediário e o refinamento completo da árvore. Como pode se observar pelos gráficos os valores intermediários aparecem em pequenos intervalos de valores combinados de *alpha* e *épsilon* e são somente esses valores que são de interesse para o projeto em questão. Os casos de nenhum nó refinado e de todos refinados não são aplicados ao projeto, pois o primeiro não acontece nada com a malha e o segundo a malha toda é refinada e isso é enviável para a pesquisa.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Blitz++ user's guide, 2009. <http:http://www.oonumerics.org/blitz/>. 10

[2] Redes neurais, 2009. <http://www.din.uem.br/ia/neurais/neural>. 9

[3] Wikipedia, 2009. <http://pt.wikipedia.org/>. 10

[4] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse. *Weka Manual*. The University of Waikato, 2009. 14

[5] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan. *Data Mining A Knowledge Discovery Approach*. Springer, 2007. 10

[6] M. O. Domingues, S. M. Gomes, and M. A. Diaz. *Análise Wavelet na Simulação Numérica de Equações Diferenciais Parciais com Adaptabilidade Espacial.* Universidade Estadual de Campinas - Departamento de Matemática Aplicada, 2001. PhD Thesis. 5

[7] S. Navega. *Princípios Essencias do Data Mining*. Infoimagem 2002 Cenadem, novembro 2002. 10

[8] R. Santos. *Princípios e Aplicações de Mineração de Dados*. 2009. Notas de Aula. 8

[9] B. Stroustup. *The C++ Programming Language*. Addison-Wesley, 1995. 9

APÊNDICE A

Arquivo dos dados de Saída Tabelados

Arquivo gerado pelo programa desenvolvido em linguagem de programação C++ que apresenta de forma tabelada os dados de saída para a realização da mineração dos dados.

```
Alfa Level Quant Nodes Epsilon Tipo

100
5 1364 0.005 1
5 0 0.055 1
5 0 0.105 1
5 0 0.155 1
5 0 0.205 1
5 0 0.255 1
5 0 0.305 1
5 0 0.355 1
5 0 0.405 1
5 0 0.455 1
5 0 0.505 1
5 0 0.555 1
5 0 0.605 1
5 0 0.655 1
5 0 0.705 1
5 0 0.755 1
5 0 0.805 1
5 0 0.855 1
5 0 0.905 1
5 0 0.955 1
5 0 1.005 1
5 0 1.055 1
5 0 1.105 1
5 0 1.155 1
5 0 1.205 1
5 0 1.255 1
5 0 1.305 1
5 0 1.355 1
```

```
5 0 1.405 1
5 0 1.455 1
5 0 1.505 1
5 0 1.555 1
5 0 1.605 1
5 0 1.655 1
5 0 1.705 1
5 0 1.755 1
5 0 1.805 1
5 0 1.855 1
5 0 1.905 1
5 0 1.955 1
5 0 2.005 1
5 0 2.055 1
5 0 2.105 1
5 0 2.155 1
5 0 2.205 1
5 0 2.255 1
5 0 2.305 1
5 0 2.355 1
5 0 2.405 1
5 0 2.455 1

105
5 1364 0.005 1
5 0 0.055 1
5 0 0.105 1
5 0 0.155 1
5 0 0.205 1
5 0 0.255 1
5 0 0.305 1
5 0 0.355 1
5 0 0.405 1
5 0 0.455 1
5 0 0.505 1
5 0 0.555 1
5 0 0.605 1
```

```
5 0 0.655 1
5 0 0.705 1
5 0 0.755 1
5 0 0.805 1
5 0 0.855 1
5 0 0.905 1
5 0 0.955 1
5 0 1.005 1
5 0 1.055 1
5 0 1.105 1
5 0 1.155 1
5 0 1.205 1
5 0 1.255 1
5 0 1.305 1
5 0 1.355 1
5 0 1.405 1
5 0 1.455 1
5 0 1.505 1
5 0 1.555 1
5 0 1.605 1
5 0 1.655 1
5 0 1.705 1
5 0 1.755 1
5 0 1.805 1
5 0 1.855 1
5 0 1.905 1
5 0 1.955 1
5 0 2.005 1
5 0 2.055 1
5 0 2.105 1
5 0 2.155 1
5 0 2.205 1
5 0 2.255 1
5 0 2.305 1
5 0 2.355 1
5 0 2.405 1
5 0 2.455 1
```

. . . . . .
. . . . . .

APÊNDICE B

Arquivo .ARFF

Arquivo .ARFF criado com os dados gerados pelo programa desenvolvido em liguagem de programação C++.

```
%1. Title: Equations Database
%2. Marilyn Menecucci Ibanez
%
@RELATION equacao

@ATTRIBUTE alfa NUMERIC
@ATTRIBUTE level NUMERIC
@ATTRIBUTE node NUMERIC
@ATTRIBUTE epsilon NUMERIC
@ATTRIBUTE class {1,2}

@DATA
40,5,1364,0.005,1
40,5,72,0.055,1
40,5,0,0.105,1
40,5,0,0.155,1
40,5,0,0.205,1
40,5,0,0.255,1
40,5,0,0.305,1
40,5,0,0.355,1
40,5,0,0.405,1
40,5,0,0.455,1
40,5,0,0.505,1
40,5,0,0.555,1
40,5,0,0.605,1
40,5,0,0.655,1
40,5,0,0.705,1
40,5,0,0.755,1
40,5,0,0.805,1
40,5,0,0.855,1
40,5,0,0.905,1
40,5,0,0.955,1
```

```
40,5,0,1.005,1
40,5,0,1.055,1
40,5,0,1.105,1
40,5,0,1.155,1
40,5,0,1.205,1
40,5,0,1.255,1
40,5,0,1.305,1
40,5,0,1.355,1
40,5,0,1.405,1
40,5,0,1.455,1
40,5,0,1.505,1
40,5,0,1.555,1
40,5,0,1.605,1
40,5,0,1.655,1
40,5,0,1.705,1
40,5,0,1.755,1
40,5,0,1.805,1
40,5,0,1.855,1
40,5,0,1.905,1
40,5,0,1.955,1
40,5,0,2.005,1
40,5,0,2.055,1
40,5,0,2.105,1
40,5,0,2.155,1
40,5,0,2.205,1
40,5,0,2.255,1
40,5,0,2.305,1
40,5,0,2.355,1
40,5,0,2.405,1
40,5,0,2.455,1
. . . . . .
. . . . . .
. . . . . .
2010,5,1364,0.005,2
2010,5,1364,0.055,2
2010,5,1364,0.105,2
2010,5,1364,0.155,2
```

```
2010,5,1364,0.205,2
2010,5,0,0.255,2
2010,5,0,0.305,2
2010,5,0,0.355,2
2010,5,0,0.405,2
2010,5,0,0.455,2
2010,5,0,0.505,2
2010,5,0,0.555,2
2010,5,0,0.605,2
2010,5,0,0.655,2
2010,5,0,0.705,2
2010,5,0,0.755,2
2010,5,0,0.805,2
2010,5,0,0.855,2
2010,5,0,0.905,2
2010,5,0,0.955,2
2010,5,0,1.005,2
2010,5,0,1.055,2
2010,5,0,1.105,2
2010,5,0,1.155,2
2010,5,0,1.205,2
2010,5,0,1.255,2
2010,5,0,1.305,2
2010,5,0,1.355,2
2010,5,0,1.405,2
2010,5,0,1.455,2
2010,5,0,1.505,2
2010,5,0,1.555,2
2010,5,0,1.605,2
2010,5,0,1.655,2
2010,5,0,1.705,2
2010,5,0,1.755,2
2010,5,0,1.805,2
2010,5,0,1.855,2
2010,5,0,1.905,2
2010,5,0,1.955,2
2010,5,0,2.005,2
```

```
2010,5,0,2.055,2
2010,5,0,2.105,2
2010,5,0,2.155,2
2010,5,0,2.205,2
2010,5,0,2.255,2
2010,5,0,2.305,2
2010,5,0,2.355,2
2010,5,0,2.405,2
2010,5,0,2.455,2
```