ISSN 2176-5480

22nd International Congress of Mechanical Engineering (COBEM 2013)
November 3-7, 2013, Ribeirão Preto, SP, Brazil
Copyright © 2013 by ABCM

# AN OPTIMAZED FORMULATION OF THE FINITE DIFERENTS METHOD IN THE FREQUENCY DOMAIN APPLIED TO ACOUSTIC WAVE PROBLEMS

**Raul Bernardo Vidal Pessolani**
**Bruno Jorge Macedo dos Santos**
**Jean Carlos Dias**
Universidade Federal Fluminense
Departamento de Engenharia Mecânica
Rua Passo da Pátria 156 sala 109 Bloco E
CEP 24210-240 Niterói - Rio de Janeiro
e-mail raul@vm.uff.br

**Marco Antonio Cetale Santos**
Universidade Federal Fluminense
Departamento de Geologia e Geofisica
Rua Passo da Pátria 156 sala 109 Bloco E
CEP 24210-240 Niterói - Rio de Janeiro
e-mail marcocetale@id.uff.br

*Abstract. - This article presents an optimized formulation applied to the acoustic waves calculation in frequency domain using Finites Differences Methods (FDM). First of all, shows the governing equations, the formulation of FDM with the discretized domain and the resolution of the system based in the LU decomposition. Next a first optimization that works with the diagonal band and transforms the original matrix in a rectangular one is implemented, developing news triangulation algorithms. A second optimization is done removing zeros still left in the main diagonal band. In the tests is shown an example with the comparison of the memory gain, computational time and reduced memory between the standard formulation and the first and the second optimizations. Finally show up the computational gains in time and memory to the calculation of the inverse matrix. It can be noted how the optimizations made the problem much more active and possible to be calculated using less memory and computational time.*

***Keywords:*** *Finite Difference Frequency Method, Acoustic wave, Geophysics.*

## 1. INTRODUTION

The Finite Difference Method is widely used to solve acoustic problems in the frequency domain due to its simplicity, practical and easy formulation. However, the size of the generated matrices and the computational time grows exponentially with the mesh. Also, this disadvantages grow when this technique is applied to inverse problems in geophysical to determine the subsurfaces, in order to study the existence of oil reservoirs (Pratt, R.G. and M. H. Worthington, 1990). Some strategies are adopted to minimize these disadvantages.

This article presents numerical algorithms to solve the wave acoustic problem in frequency staggered-grid finite difference scheme. Our main motivation is to reduce the size of the system solver and the CPU time. Our goal is to provide an optimized formulation of the Finite Difference Methods in the frequency domain, adopting two optimizations that transform the original square and sparse matrix, in a rectangular one. Zeros are not stored. Some optimized algorithms are developed from solvers of LU decomposition to enhance the existing methods in order to minimize the errors associated with, improving the quality of the numerical results and reducing its computational effort. Also a non reflecting boundary technique is applied to reduce the side effects. Clayton and Engquist (1977) proposed the Absorbing Boundary Condition (ABC) technique by applying a one-way wave equation in the boundary region, which proved to be efficient for events not at shallow angles on the contour. The ABC method is applied and optimized aiming to reduce wave reflection at the borders, with results shown in terms of the total energy for “infinite” and nonreflecting models for varying absorbing layers. Results are shown in terms of the time and the size of the matrix comparing with the standard one.

First the acoustic wave problem formulation is presented and also the ABC Boundary non reflecting method. In addition, the Finite Difference equations are deduced and the system is showed. Later, the first and second optimization in order to reduces the size of the stored matrix and the time computation are presented. Finally through two examples one can see the percentage of the gain for memory and computation comparing with the standard method.

Some examples show the memory and computational gain obtained.

## 2. PROBLEM FORMULATION

The wave equation in time domain in two dimensions is written below :

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}$$ (1)

where *p* is the pressure field and *c* is the speed of pressure wave like space function.

To solve it in the frequency domain, the Fourier Transform (Kreyszig (1993)) must be used in the above equation to pass it from the time domain to the frequency domain. Thus, one obtains (Durran 1989):

$$\frac{w^2}{c^2}p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial p^2}{\partial y^2}$$ (2)

Where: $w = 2\pi f$.

The equation can be rearranged replacing the wave number $k^2 = \frac{w^2}{c^2}$ and adding source term $S$. Thus, one obtains the following equation:

$$k^2 p - \left[\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right] = -S$$ (3)

Finding the solution of the equation above without reflections at the boundaries, it used the follow absorption Boundary, called ABC (*Absorving Boundary Condiction)* (Clayton and Engquist (1979)):

$$\frac{\partial p}{\partial n} - ikp = 0$$ (4)

Where *n* is the normal direction to the edge that applies the condition and *i* is the unit complex.

## 3. FINITE DIFFERENCES METHODS AND SPATIAL DISCRETIZATION

To solve the Helmholtz equation by the FDM, it´s necessary to discretize the domain with *Nx* points in the *x* direction uniformly distant of *Dx* and *Ny* points in the *y* direction uniformly distant of *Dy*, as shown in Fig. 1 (Ajo-Franklin (2005)):

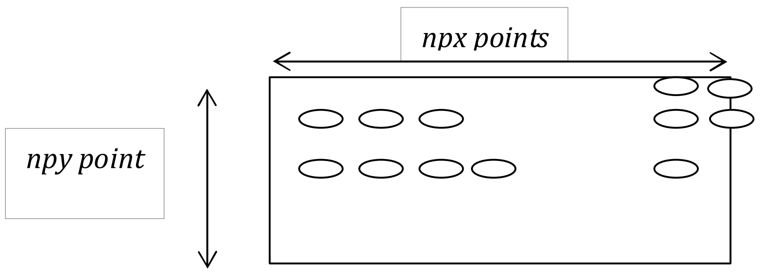

Figure 1 - Rectangular domain discretized by FDM.

The second derivate in relation to the x directed is approximated by the expression:

$$\frac{\partial^2 p}{\partial x^2} = \frac{p_{i-1,j} - 2p_{i,j} + p_{i+1,j}}{\Delta x^2}$$ (5)

that $\Delta x$ is the spacing in $x$ direction.

Using this approximation it is possible to rewrite the Helmholtz equation as follows:

$$k^2 p_{i,j} + \left[\frac{p_{i-1,j}-2p_{i,j}+p_{i+1,j}}{\Delta x^2}\right] + \left[\frac{p_{i,j-1}-2p_{i,j}+p_{i,j+1}}{\Delta y^2}\right] = -S_{i,j} \tag{6}$$

With the derivatives in the ABC equation, the equation to the boundaries of the domain is expressed as:

$$\frac{p_{i+1,j}-p_{i,j}}{\Delta x^2} - ikp_{i,j} \tag{7}$$

So, assuming a 4 x 4 domain, a matrix is generated (see Fig. 2):

$$\begin{bmatrix}
Xi_{1,1} & X_{1,2} & & & X_{1,5} & & & & & & & & & & & \\
 & Xi_{2,2} & & & & X_{2,6} & & & & & & & & & & \\
 & & Xi_{3,3} & & & & X_{3,7} & & & & & & & & & \\
 & & X_{4,3} & Xi_{4,4} & & & & X_{4,8} & & & & & & & & \\
 & & & & Xi_{5,5} & X_{5,6} & & & & & & & & & & \\
 & X_{6,2} & & & X_{6,5} & X_{6,6} & X_{6,7} & & & X_{6,10} & & & & & & \\
 & & X_{7,3} & & & X_{7,6} & X_{7,7} & X_{7,8} & & & X_{7,11} & & & & & \\
 & & & & & & X_{8,7} & Xi_{8,8} & & & & & & & & \\
 & & & & & & & & Xi_{9,9} & X_{9,10} & & & & & & \\
 & & & & & X_{10,6} & & & X_{10,9} & X_{10,10} & X_{10,11} & & & X_{10,14} & & \\
 & & & & & & X_{11,7} & & & X_{11,10} & X_{11,11} & X_{11,12} & & & X_{11,15} & \\
 & & & & & & & & & & X_{12,11} & Xi_{12,12} & & & & \\
 & & & & & & & & X_{13,9} & & & & Xi_{13,13} & X_{13,14} & & \\
 & & & & & & & & & X_{14,10} & & & & Xi_{14,14} & & \\
 & & & & & & & & & & X_{15,11} & & & & Xi_{15,15} & \\
 & & & & & & & & & & & X_{16,12} & & & X_{16,15} & Xi_{16,16}
\end{bmatrix}$$

Figure 2 - Coefficients matrix generated by the Finite Difference Method

that $X$ refers to real numbers and $X_i$ to complex numbers, due to application of ABC.

Note that the above matrix is a sparse matrix with the order of (*npx.npy, anpx.npy*). The bandwidth is (*2npx+1*) size, for the node´s domain relates with its neighboring, the right and left and the upper and lower ones, according to the diagram showed in Fig. 3:

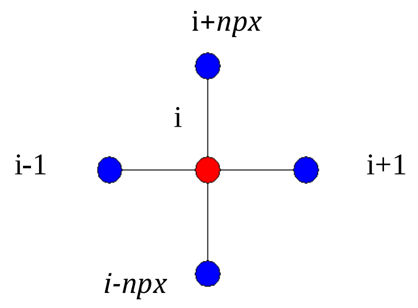

Figure 3 - Relationship of node *i* with its neighbors

The complex numbers $X_i$ come from the application of ABC technique. The frequency appears in the diagonal, só that, changing the frequency, just the diagonal is changed.

Assembling to a desired frequency, the coefficients' matrix **A** and the independent terms' vector **b**, form a system. The vector b represents the source point. A common technique to compute inverse problems is to vary the source position, and solving the system repeated times. The matrix A remains constant.

In Inverse geophysics problems, the system (8) has to be solved many times for different values of **b**, without changing **A**. For this purpose LU factorization is indicated (Press et alii (2007)), which transforms the original matrix **A** in a upper triangular matrix and lower triangular matrix (9) as follows:

**A.x = b** (8)

**A = L.U** (9)

**L.U.x = b**
(10)

**L.y = b**
(11)

**U.x = y**
(12)

The triangularization of **A** is made just once. The value of b are the columns of the Identity matrix, computing each column of the Inverse matrix. The system is quickly solved by backward (12) and forward substitution (11).

## 4. FIRST OPTIMIZATION

The matrix **A** is sparse and more then 90% of its elements are zeros Thus, in order to optimized the procedure a first otimization was done to reduce the space memory and the computational time required, and consists in store the original square matrix in a rectangular reduced one of the order (*npx.npy, 2npx+1*), and eliminating the zero terms, as follow system (see fig.4)

$$\begin{bmatrix}
 & & & Xi_{1,1} & X_{1,2} & & X_{1,5} \\
 & & & Xi_{2,2} & & & X_{2,6} \\
 & & & Xi_{3,3} & & & X_{3,7} \\
 & & X_{4,3} & Xi_{4,4} & & & X_{4,8} \\
 & & & Xi_{5,5} & X_{5,6} & & \\
X_{6,2} & & X_{6,5} & X_{6,6} & X_{6,7} & & X_{6,10} \\
X_{7,3} & & X_{7,6} & X_{7,7} & X_{7,8} & & X_{7,11} \\
 & & X_{8,7} & Xi_{8,8} & & & \\
 & & & Xi_{9,9} & X_{9,10} & & \\
X_{10,6} & & X_{10,9} & X_{10,10} & X_{10,11} & & X_{10,14} \\
X_{11,7} & & X_{11,10} & X_{11,11} & X_{11,12} & & X_{11,15} \\
 & & X_{12,11} & Xi_{12,12} & & & \\
X_{13,9} & & & Xi_{13,13} & X_{13,14} & & \\
X_{14,10} & & & Xi_{14,14} & & & \\
X_{15,11} & & & Xi_{15,15} & & & \\
X_{16,12} & & & Xi_{16,16} & & &
\end{bmatrix}$$

Figure 4 - Reduced matrix after first optimization

Note that the main diagonal was stored in the middle column and the original columns are now on the diagonal of the reduced matrix.

Then, a triangulation algorithm that works with the reduced matrix was implemented. The results of the application of these algorithms is a matrix in order of (*npx.npy,2npx+1*), also reduced, and has the terms of the matrices **L** and **U** respectivaly, as one can see on fig. 5:

ISSN 2176-5480

$$\begin{bmatrix}
 & & & & U_{1,1} & U_{1,2} & U_{1,3} & U_{1,4} & U_{1,5} \\
 & & & L_{2,1} & U_{2,2} & U_{2,3} & U_{2,4} & U_{2,5} & U_{2,6} \\
 & & L_{3,1} & L_{3,2} & U_{3,3} & U_{3,4} & U_{3,5} & U_{3,6} & U_{3,7} \\
 & L_{4,1} & L_{4,2} & L_{4,3} & U_{4,4} & U_{4,5} & U_{4,6} & U_{4,7} & U_{4,8} \\
L_{5,1} & L_{5,2} & L_{5,3} & L_{5,4} & U_{5,5} & U_{5,6} & U_{5,7} & U_{5,8} & U_{5,9} \\
L_{6,2} & L_{6,3} & L_{6,4} & L_{6,5} & U_{6,6} & U_{6,7} & U_{6,8} & U_{6,9} & U_{6,10} \\
L_{7,3} & L_{7,4} & L_{7,5} & L_{7,6} & U_{7,7} & U_{7,8} & U_{7,9} & U_{7,10} & U_{7,11} \\
L_{8,4} & L_{8,5} & L_{8,6} & L_{8,7} & U_{8,8} & U_{8,9} & U_{8,10} & U_{8,11} & U_{8,12} \\
L_{9,5} & L_{9,6} & L_{9,7} & L_{9,8} & U_{9,9} & U_{9,10} & U_{9,11} & U_{9,12} & U_{9,13} \\
L_{10,6} & L_{10,7} & L_{10,8} & L_{10,9} & U_{10,10} & U_{10,11} & U_{10,12} & U_{10,13} & U_{10,14} \\
L_{11,7} & L_{11,8} & L_{11,9} & L_{11,10} & U_{11,11} & U_{11,12} & U_{11,13} & U_{11,14} & U_{11,15} \\
L_{12,8} & L_{12,9} & L_{12,10} & L_{12,11} & U_{12,12} & U_{12,13} & U_{12,14} & U_{12,15} & U_{12,16} \\
L_{13,9} & L_{13,10} & L_{13,11} & L_{13,12} & U_{13,13} & U_{13,14} & U_{13,15} & U_{13,16} & \\
L_{14,10} & L_{14,11} & L_{14,12} & L_{14,13} & U_{14,14} & U_{14,15} & U_{14,16} & & \\
L_{15,11} & L_{15,12} & L_{15,13} & L_{15,14} & U_{15,15} & U_{15,16} & & & \\
L_{16,12} & L_{16,13} & L_{16,14} & L_{16,15} & U_{16,16} & & & &
\end{bmatrix}$$

Figure 5 - Matrix that contains the matrices L and U.

Even reducing the size of the matrix, one can see that some columns of A are zeros. A second optimization has to be made in an reduced inverse matrix, removing all the columns formed by zeros, in consequence, the obtained matrix, with just 5 columns and 2.*npx*+1 lines.

## 5. TESTS

A program was implemented in Fortran 90, that incorporates algorithm for the solution of the sparse reduced matrices and compares with the standard LU factorization algorithm. The processing time was measured and the numbers of operations was evaluated and compared.

Table 1 shows the amount of memory occupied. On the first column, one can see the size of the domain, and on the second the size of the entire standard matrix. The third and fourth columns show the size of the matrix using the first and second optimization and then three last columns shows the gain comparing the various the standard, first and second optimizations. One can see the percentage of the space memory reduced by the optimizations.

Table 1: Amount of memory dispended and the space of memory gain by the optimizations.

| **Domain** | **Matrix size Standard (Str)** | **Size of the 1th opt** | **Size of the 2nd opt** | **standard vs 1th opt (%)** | **1th vs 2nd opt (%)** | **2nd vs standard (%)** |
|---|---|---|---|---|---|---|
| **5** | 25x25 | 25x11 | 25x5 | **56** | 54,55 | **80,00** |
| **10** | 100X100 | 100x21 | 100x5 | **79** | 76,19 | **95,00** |
| **15** | 225x225 | 225x31 | 225x5 | **86,22** | 83,87 | **97,78** |
| **20** | 400x400 | 400x41 | 400x5 | **89,75** | 87,80 | **98,75** |
| **25** | 625x625 | 625x51 | 625x5 | **91,84** | 90,20 | **99,20** |
| **30** | 900x900 | 900x61 | 900x5 | **93,22** | 91,80 | **99,44** |
| **35** | 1225x1225 | 1225x71 | 1225x5 | **94,20** | 92,96 | **99,59** |
| **40** | 1600x1600 | 1600x81 | 1600x5 | **94,94** | 93,83 | **99,69** |
| **45** | 2025x2025 | 2025x91 | 2025x5 | **95,51** | 94,51 | **99,75** |
| **50** | 2500x2500 | 2500x101 | 2500x5 | **95,96** | 95,05 | **99,80** |
| **55** | 3025x3025 | 3025x111 | 3025x5 | **96,33** | 95,50 | **99,83** |
| **60** | 3600x3600 | 3600x121 | 3600x5 | **96,64** | 95,87 | **99,86** |
| **65** | 4225x4225 | 4225x131 | 4225x5 | **96,90** | 96,18 | **99,88** |
| **70** | 4900x4900 | 4900x141 | 4900x5 | **97,12** | 96,45 | **99,90** |
| **75** | 5625x5625 | 5625x151 | 5625x5 | **97,32** | 96,69 | **99,91** |
| **80** | 6400x6400 | 6400x161 | 6400x5 | **97,48** | 96,89 | **99,92** |
| **85** | 7225x7225 | 7225x171 | 7225x5 | **97,63** | 97,08 | **99,93** |
| **90** | 8100x8100 | 8100x181 | 8100x5 | **97,76** | 97,24 | **99,94** |
| **95** | 9025x9025 | 9025x191 | 9025x5 | **97,88** | 97,38 | **99,94** |

| | | | | | | |
|---|---|---|---|---|---|---|
| **100** | 10000x10000 | 10000x201 | 10000x5 | **97,99** | 97,51 | **99,95** |
| **105** | 11025x11025 | 11025x211 | 11025x5 | **98,09** | 97,63 | **99,95** |
| **110** | 12100x12100 | 12100x221 | 12100x5 | **98,17** | 97,74 | **99,96** |

Table 2 shows the processing time in seconds. On the second column, one can see the time required to solve the standard method, and on the third and fourth columns, the computational time for the first and second optimization. The last two columns show a comparison between the first and second optimization with the standard end from each other. It can be seen that the gain increases exponentially and, for example, for a domain with 110x110 nodes, the 2nd optimization is 99,81% faster than the standard on.

Table 2: Processing Time with the comparison of the computational time gain.

| Domain | Str time (s) | Time 1th opt | Time 2nd opt | Str vs 2nd opt | 1th vs 2nd opt |
|---|---|---|---|---|---|
| **5** | 0 | 0 | 0 | 91,67 | 0,00 |
| **10** | 0 | 0 | 0 | 97,47 | 0,00 |
| **15** | 0 | 0 | 0 | 98,14 | 0,00 |
| **20** | 0,03 | 0 | 0 | 98,75 | 0,00 |
| **25** | 0,12 | 0,01 | 0,01 | 99,19 | 33,33 |
| **30** | 0,79 | 0,02 | 0,02 | 99,37 | 41,67 |
| **35** | 3,22 | 0,06 | 0,04 | 99,47 | 26,32 |
| **40** | 9,6 | 0,12 | 0,07 | 99,53 | 29,03 |
| **45** | 23,4 | 0,19 | 0,14 | 99,59 | 33,33 |
| **50** | 49,54 | 0,31 | 0,22 | 99,64 | 36,71 |
| **55** | 95,46 | 0,51 | 0,34 | 99,67 | 33,62 |
| **60** | 168,48 | 0,79 | 0,50 | 99,70 | 32,93 |
| **65** | 282,66 | 1,16 | 0,77 | 91,67 | 32,17 |
| **70** | 451,43 | 1,64 | 1,10 | 97,47 | 30,99 |
| **75** | 700,6 | 2,3 | 1,56 | 98,14 | 40,20 |
| **80** | 1051,02 | 3,13 | 2,16 | 98,75 | 28,12 |
| **85** | 1534,55 | 4,90 | 2,93 | 99,68 | 26,86 |
| **90** | 2195,9 | 5,37 | 3,86 | 99,76 | 27,28 |
| **95** | 3094,34 | 7,00 | 5,12 | 99,77 | 26,88 |
| **100** | 4221,64 | 8,98 | 6,53 | 99,79 | 27,12 |
| **105** | 5700,11 | 11,42 | 8,35 | 99,80 | 33,33 |
| **110** | 7497,49 | 14,38 | 10,48 | 99,81 | 41,67 |

Next, Figure 6 shows how the computational gain grows up with the order of the system (Figure 6). Note that how much higher is the order, or the number of the points in the domain, higher is the memory and CPU time saving (Figure 7). So one can conclude that the use of algorithm to the reduced matrix is almost mandatory to optimize computational resources.

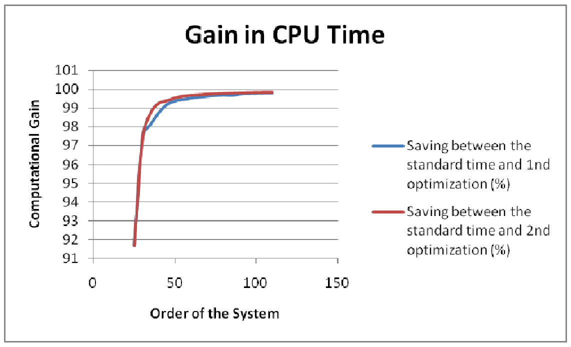

Figure 6: Computational gain

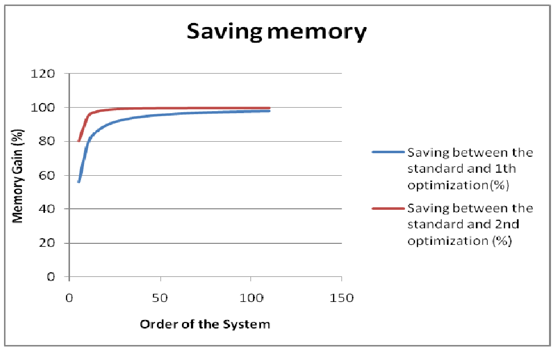

Figure 7: Memory gain.

Finally, Table 3 shows the numbers of operations for each algorithm. Note that the use of the optimizations gives a significant reducing for the numbers of operations. For example, in a domain with 110x110 points, the number of operations for the 2nd optimization is 3993 times lesser than the standard

Table 3: Comparison of the operation numbers gain

| Domain | Full Matrix Order | Sparse matrix order | N°. operations Standard form | N°. operations 2nd optimization | Total gain |
|---|---|---|---|---|---|
| 5 | 25x25 | 25x11 | 5274 | 800 | 6,59 |
| 10 | 100X100 | 100x21 | 3,33 x10$^5$ | 1,13 x10$^4$ | 29,47 |
| 15 | 225x225 | 225x31 | 3,79 x10$^6$ | 5,52 x10$^4$ | 150,40 |
| 20 | 400x400 | 400x41 | 2,13 x10$^7$ | 1,71 x10$^5$ | 124,56 |
| 25 | 625x625 | 625x51 | 8,14 x10$^7$ | 4,12 x10$^5$ | 193,35 |
| 30 | 900x900 | 900x61 | 2,43 x10$^8$ | 8,46 x10$^5$ | 287,23 |
| 35 | 1225x1225 | 1225x71 | 6,13 x10$^8$ | 1,56 x10$^6$ | 392,95 |
| 40 | 1600x1600 | 1600x81 | 1,37 x10$^9$ | 2,65 x10$^6$ | 516,98 |
| 45 | 2025x2025 | 2025x91 | 2,77 x10$^9$ | 4,22 x10$^6$ | 656,40 |
| 50 | 2500x2500 | 2500x101 | 5,21 x10$^9$ | 6,42 x10$^6$ | 811,53 |
| 55 | 3025x3025 | 3025x111 | 9,23 x10$^9$ | 9,38 x10$^6$ | 984,01 |
| 60 | 3600x3600 | 3600x121 | 1,56 x10$^{10}$ | 1,32 x10$^7$ | 1181,82 |
| 65 | 4225x4225 | 4225x131 | 2,51 x10$^{10}$ | 1,82 x10$^7$ | 1379,12 |
| 70 | 4900x4900 | 4900x141 | 3,92 x10$^{10}$ | 2,45 x10$^7$ | 1600,00 |
| 75 | 5625x5625 | 5625x151 | 5,93 x10$^{10}$ | 3,22 x10$^7$ | 1841,61 |
| 80 | 6400x6400 | 6400x161 | 8,74 x10$^{10}$ | 4,16 x10$^7$ | 2100,96 |
| 85 | 7225x7225 | 7225x171 | 1,26 x10$^{11}$ | 5,30 x10$^7$ | 2377,36 |
| 90 | 8100x8100 | 8100x181 | 1,77 x10$^{11}$ | 6,66 x10$^7$ | 2657,66 |
| 95 | 9025x9025 | 9025x191 | 2,45 x10$^{11}$ | 8,26 x10$^7$ | 2966,10 |
| 100 | 10000x10000 | 10000x201 | 3,33 x10$^{11}$ | 1,01 x10$^8$ | 3297,03 |
| 105 | 11025x11025 | 11025x211 | 4,47 x10$^{11}$ | 1,23 x10$^8$ | 3634,15 |
| 110 | 12100x12100 | 12100x221 | 5,91 x10$^{11}$ | 1,48 x10$^8$ | 3993,24 |

In figure 3, one can see how the operation number's gain grows up with the numbers of points adopted.

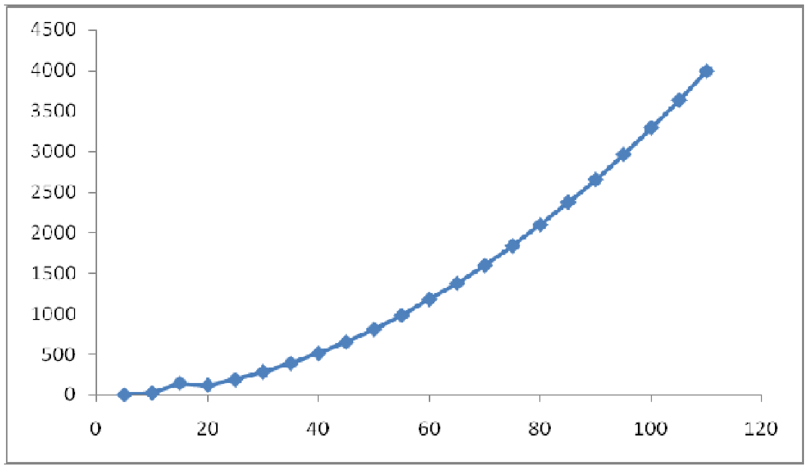

Figure 3: Numbers of operation gains in function of numbers of points *npx*.

## 6. CALCULATION OF THE INVERSE

The inverse of a Matrix can be calculated using **L.U** changing for each system the independent term **b**. On each iteration, each b will be the columns of the identity matrix. The inverse of the sparse and banded matrix will not be a sparse and banded, but a full one.

Using algorithm that works with the reduced matrix, one can reduce significantly the computational time. Table 4 shows a comparison between the formulations, and on Figure 8 one can see a graph showing the computational gain obtained using the solver with the 2nd optimization.

Table 4: Comparison of computational time gain to the inverse calculation

| Nodes | CPU Time Standard (s) | CPU Time Opt | Computational gain |
|---|---|---|---|
| 5 | 0 | 0 | 0 |
| 10 | 0,01 | 0 | 0 |
| 15 | 0,01 | 0,01 | 1 |
| 20 | 0,07 | 0,03 | 2,33 |
| 25 | 0,4 | 0,09 | 4,44 |
| 30 | 2,75 | 0,2 | 13,75 |
| 35 | 9,41 | 0,38 | 24,76 |
| 40 | 23,82 | 0,66 | 36,09 |
| 45 | 51,6 | 1,7 | 30,35 |
| 50 | 101,59 | 1,74 | 58,38 |
| 55 | 185,99 | 2,54 | 73,22 |
| 60 | 323,08 | 3,89 | 83,05 |
| 65 | 531,57 | 5,28 | 100,67 |
| 70 | 836,47 | 7,62 | 109,77 |
| 75 | 1284,96 | 11,1 | 115,76 |
| 80 | 1960,15 | 12,88 | 152,18 |
| 85 | 2799,73 | 19,25 | 145,44 |
| 90 | 3943,94 | 26,99 | 146,12 |

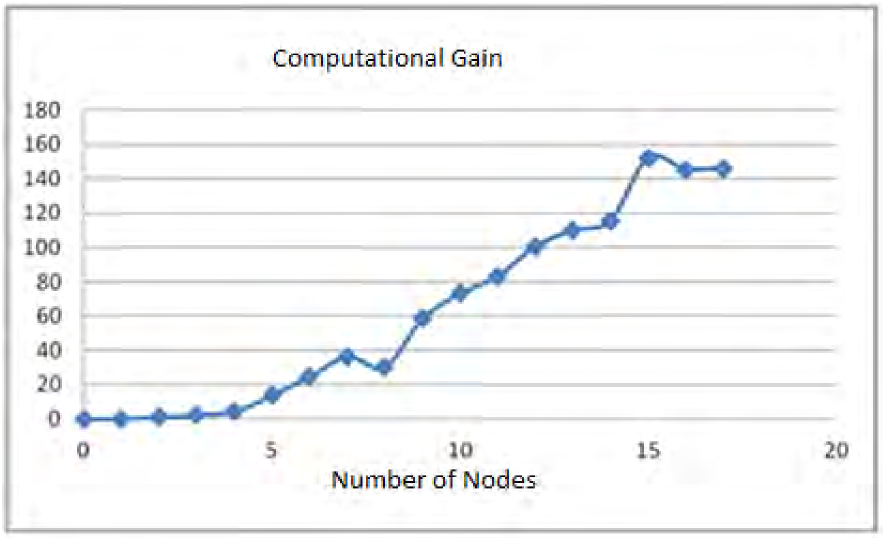

Figure 8 – Computational gain Increase  in function of adopted number's node

## 7. CONCLUSIONS

A Finite Difference Frequency formulation was presented to solve wave acoustic propagation. Two optimizations to reduce both storing space memory and computational time were made. Also a classical nonreflecting boundary method – ABC – was applied aiming to reduce wave reflections at the borders of the 2D computational domain.

Through examples one can see that a great saving memory and computational time was obtained by reducing the computational time and memory space up to 20 times. This indicates, that this optimizations are strongly recommended to solve the Finite Difference Frequency Domain problem, reducing the memory space and also the computational time, making possible to solve these problems in small computers.

The decrease of computational time and great saving memory is results of a decrease of numerical operations given by the second optimization, so, the numerical errors fall significantly, resulting in a more accurate result then the

ISSN 2176-5480

coventionals ones.

Some applications can be observed, the authors developed these optimizations to model numerically the situations of probes sent to soil and detect oil and tectonic plates with more precision. But these algorithms can be used for spring mass system, electrical circuits, analysis of lattice and every situation involving linear sparse systems.

## 8. ACKNOWLEDGMENTS

The authors from UFF acknowledge the financial support through PETROBRAS.

## 9. REFERENCES

Ajo-Franklin, J.B., 2005, Frequency-domain modeling for the scalar wave equation: an introduction: Earth resources laboratory Laboratory Industry Consortia Technical Reports. Massachusetts Institute of Technology.

Clayton, R. and Engquist, B., Absorbing boundary conditions for acoustic and elastic wave equation. *Bull. Seis. Am.,* v.67, pp.1529-1540, 1977.

Durran, D.R., *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics*. New York: Springer-Verlag, 1999.

Kreyszig, E., Advanced Engineering Mathematics. Singapore: John Wiley and Sons, 1993

Pratt, R.G. and M. H. Worthington, 1990, Inverse Theory applied to multi-source cross-hole tomography, Part I: acoustic wave-equation method: Geophysical Prospecting 38, 287-310.

Press, H.W., Teukolsky, S. A., Vetterling, W. T., and Flannery B.P., *Numerical Recipes*, 3rd Edition, Ed. Cambridge University, 2007.

Chapra, Steven C., and Canale, Raymond P., *Métodos Numéricos para Engenharia*, 5th edition, Ed. McGraw-Hill, 2008.

## 10. Responsibility notice

The authors are the only responsible for the printed material included in this paper.