# Finite Difference Method applied to DML simulation. Part 1.

ChristianV (Homeswinghome@diyAudio)

10 dec 2022

**Abstract**

Simulation of Distributed Mode Loudspeaker (DML) thanks to the Finite Difference Method (FDM)

## Contents



**Disclaimer** : this paper is written in the context of DIY DML building with the target to identify some design rules to help in the panel construction. This document is not written in the context of any academic or scientific work. Its content is reviewed only by the feedback it can get while posting it in audio DIY forum like diyAudio.

## 1 Introduction

I hope this paper is the first of a series leading to the simulation of Distributed Mode Loudspeaker.

Searching in the different possibilities like FEM (Finite element Method) or electrical analogy I found the Finite Difference Method, which is a an old technique to solve differential equations as a possible tool to go to a simulation of a DML. This method started in the late 18th with a boom in the 60th before the introduction of FEM.

The advantage and the difficulty of this method is it works closely of the equations. It will be probably not as powerful as an FEM tool can be but it might be more proper to develop the knowledge of DML and the possibility to integrate all the aspects of the problem : geometry, suspension, electromechanical exciter, etc.

Here is the comment about this method in [1] §3 :

> The finite difference method, usually referred as FDM, is a numerical method that solves the equations of boundary problems using mathematical discretization. It is the oldest but still very viable numerical methods for solution of partial differential equation and hence is suitable for solving plate bending equation. This method is sufficienly accurate for thin plate analysis (Ezeh et al., 2013). It is probably the most transparent and the most general method among the various numerical approaches available for analysis of thin plates. Although, a number of finite element based software packages are available now-a-days (2017), the finite difference method is still regarded for its straightforward nature and a minimum requirement on hardware

No need of specific a software neither, Python which is a multi-plateform language has in its standard libraries (numpy, scipy. . . ) the functions to write a FDM script.

To go along all the process from the theory to results from a script, this introduction paper is limited to the simply supported and clamped boundaries conditions (see after); 2 advantages beyond that : they are the most simple conditions, formal mode expressions are known for certain conditions, tables exist for others, allowing the script debugging.

Among the different sources about FDM, Numerical Solution of the Poisson Equation Using Finite Difference Matrix Operators [2] inspired largely this paper.

## 2 Thin plate partial derivative equation (Kirchhoff-Love model)

Most (not to say all?) of the papers dealing with DML at the membrane level base their approach on the thin plate theory. From Wiki Kirchhoff–Love plate theory :

> The Kirchhoff–Love theory of plates is a two-dimensional mathematical model that is used to determine the stresses and deformations in thin plates subjected to forces and moments.

For an isotropic homogeneous material, the governing equation is :

$$D(\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4}) = -q(x,y,t) - \mu\frac{\partial^2 w}{\partial t^2} \tag{1}$$

where :

- $D$ is the bending stiffness
- $x, y$ the axis of the membrane plane
- $w$ the deflection of the plate perpendicular to its plane (following $z$ axis)
- $\mu$ the areal mass
- $\partial$ the partial derivative (meaning the derivative according to the denominator) of $w$
- $q$ the load applied in the $z$ direction to the plate

$q$ is the known value (ie the force from the exciter)

$w$ is the unknown value (the deflection).

The other elements are collected from the plate geometry dimensions and material characteristics.

For example, from $w$, the speed $v$ at each point of the membrane can be derived to compute the SPL at distance.

The eigenfrequenies (modes) and their according geometry can be also identified from this governing equations.

Some transient simulation (behavior over the time $t$) can also be derived from the main equation.

In DML building, orthotropic materials are often encountered, like the plywood. Those materials have different bending stiffness according to the directions (ie along or across the grain).

To cover that cas, a more general governing equation is used :

$$D_x\frac{\partial^4 w}{\partial x^4} + 2D_{xy}\frac{\partial^4 w}{\partial x^2 \partial y^2} + D_y\frac{\partial^4 w}{\partial y^4} = -q(x,y,t) - \mu\frac{\partial^2 w}{\partial t^2} \tag{2}$$

- $D_x$ is the bending stiffness in the $x$ axis.
- $D_y$ is the bending stiffness in the $y$ axis.
- $D_{xy}$ : to be clarified... TODO

## 3 Finite Difference Method (FDM)

The key idea of the Finite Difference Method (FDM) (see FDM wikipedia page) is to work on a discretized domain of the original continuous domain and to replace the partial derivatives by approximations based on the values of the points surrounding the point of interest.

The coefficients to apply to the neighbor points can be find for example in the Wik page "Finite difference coefficient".

For each degree of derivatives, several approximations are proposed. The usually encountered one is the approximation of degree 2.

The problem is then transformed in a linear system of equation represented in a matrix format.

The first step which is the easiest one is to mesh the membrane. We will use the most standard and probably basic shape for the mesh : a rectangle of size $\Delta x$ by $\Delta y$. Other shapes like hexagons are possible and maybe more suitable for circular membrane shape.

There is no need $\Delta x$ and $\Delta y$ being equal and constant over the membrane. In this first time this possibility is not used, $\Delta x$ and $\Delta y$ will be set equal and constant

The plate of dimensions $L_x$ by $L_y$ is divided in $N_x$ $\Delta_x$ elements in the $x$ direction with the index is $j$ in the interval $[0..N_x - 1]$ and in $N_y$ $\Delta_y$ elements in the $y$ direction with the index is $k$ in the interval $[0..N_y - 1]$.

The plate is like that divided in $N_x N_y$ elementary mesh.

The elements along the perimeter form the boundary. Due to the application to DML, no internal boundary (like the hole of a guitar top) is included here.

**See figure 1**

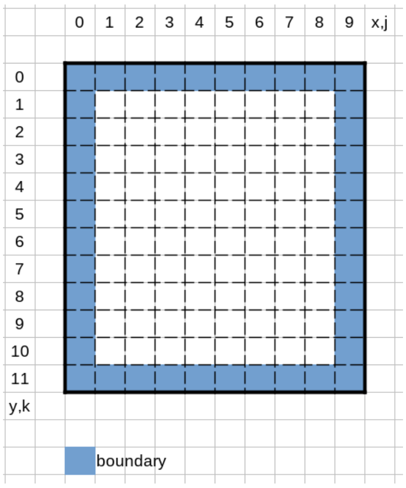


Figure 1: DML plate mesh

**Note** : one important thing to keep in mind is the 2D view of figure 1 is the geometrical view. To make the simulation, one equation per mesh (cell) will be written leading to $N_x N_y$ equations and then a $N_x N_y$ by $N_x N_y$ system matrix in front so $(N_x N_y)^2$ coefficients in the matrix. Hopefully, most of them are null. This one task of the Python script to write this matrix.

In this idea, a tool called a stencil is used to show which and how each neighbor points are combined to be embedded in the system matrix.

The stencil for plate problem so called biharmonic operator is usued for the left side of the governing equation.

The stencil shape is shown in **figure 2**.

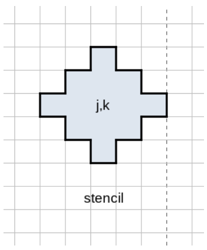


Figure 2: Biharmonic operator stencil

The grey cells show the cells that are included in the calculation of the left side of the governing equation (derivatives related to the geometry) for the mesh of coordinates $j, k$.

The **figure 3** shows the stencil in place for the mesh $2, 2$. All the meshes in the light yellow area have a stencil inside the plate geometry.

Let see in the “boundary conditions” paragraph what happens for the white or blue meshes for which cells of the stencil are out the geometry.

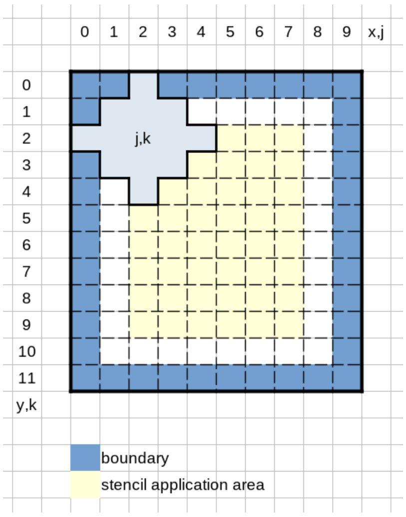


Figure 3: Stencil in 2,2

# 4 Boundary conditions

## 4.1 Bondary conditions for clamped, simply supported

Clamped, simply supported or free edge are the most common boundary conditions.

- Simply supported : the edge behaves like a hinge. The plate can move freely around the edge axis but not in the $z$ direction perpendicular to the plate.
- Clamped : the rotation around the edge is blocked like a rusty hinge.

In a mathematical wording, for the points of the edges, this is translated by :

- Clamped : $w_{k,j} = 0$, $\frac{\partial w}{\partial n} = 0$
- Simply supported : $w_{k,j} = 0$, $\frac{\partial^2 w}{\partial n^2} = 0$

$n$ being the direction perpendicular to the edge (normal).

## 4.2 Finite differences at the boundaries

The first and second derivatives are as follow in finite differences :

$$\frac{\partial w}{\partial x} \approx \frac{1}{2\Delta_x}(w_{j+1} - w_{j-1})$$

$$\frac{\partial^2 w}{\partial x^2} \approx \frac{1}{{\Delta_x}^2}(w_{j+1} - 2w_j + w_{j-1})$$

Taking the example of the left edge with $j = 0$ and $w_{0,k} = 0$, the values of the fictives nodes are known:

- Clamped : $w_{-1} = w_1$
- Simply supported : $w_{-1} = -w_1$

Then replacing the $w_{-1}$ in the generic stencil, the points just before the edge (white cells) have a specific stencil (see **figure 4**) where 1 (clamped edge) or -1 (simply supported edge) is added to the coefficient of the central cell.

With that and by setting to 1 the coefficient of each edge points, to 0 the other on the same line, in the system matrix, the boundaries are taking into account.

# 5 System matrix

I borrow from M Zaman [2] the **figure 5** for the transformation from the 2D representation to the 1D one and back.

The system matrix has to be filled as in **figure 6** (simple example of a 6 by 7 elements plate)

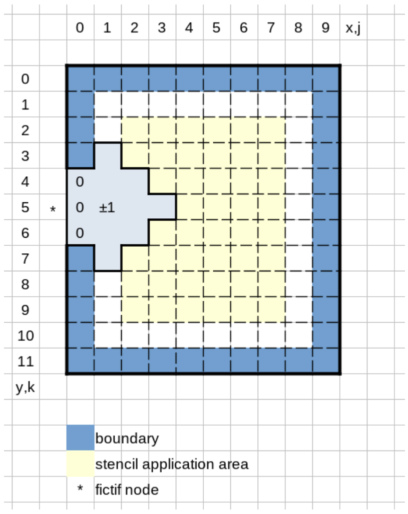


Figure 4: Edge stencil

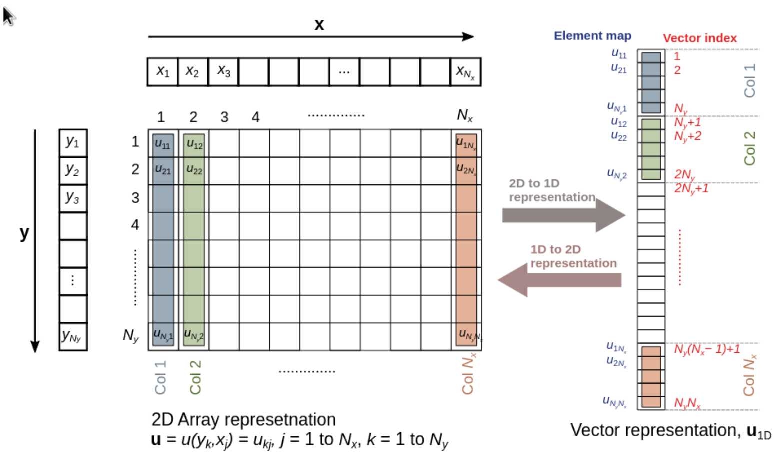


Figure 5: 2D to 1D transformation

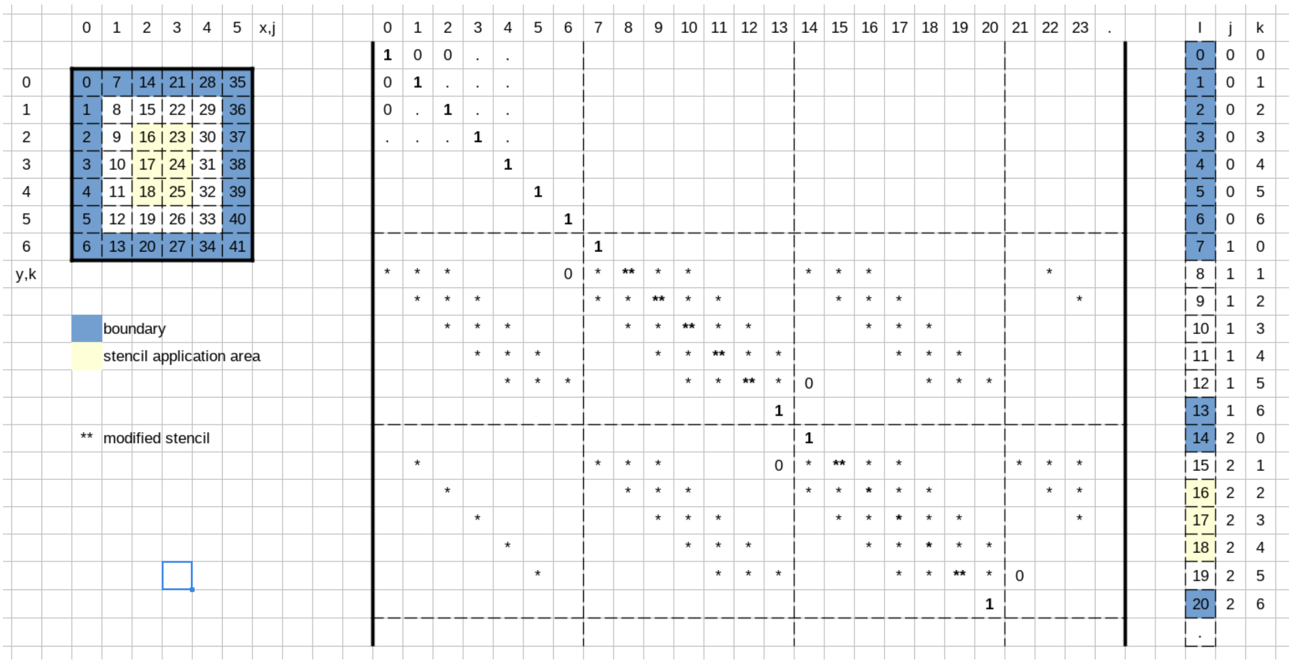


Figure 6: System matrix

## 5.1 System matrix filling

As the System Matrix is filled once per simulation independently of the type of simulation, the proposal is to choose a systematic method by loops on j (row index) and k (column index), then filling the System matrix using the index cell i.

This choice is made supposing it will be the most versatile for other boundary conditions.

$$i = k + N_y j$$

To fill the system matrix, each cell is identified belonging to one area :

- North, East, South or West (external) Bound
- North, East, South or West Internal (Bound)
- North-East, South-East, South-West, North-West (internal corners)
- Internal Nodes

According to the area, the system matrix is filled.

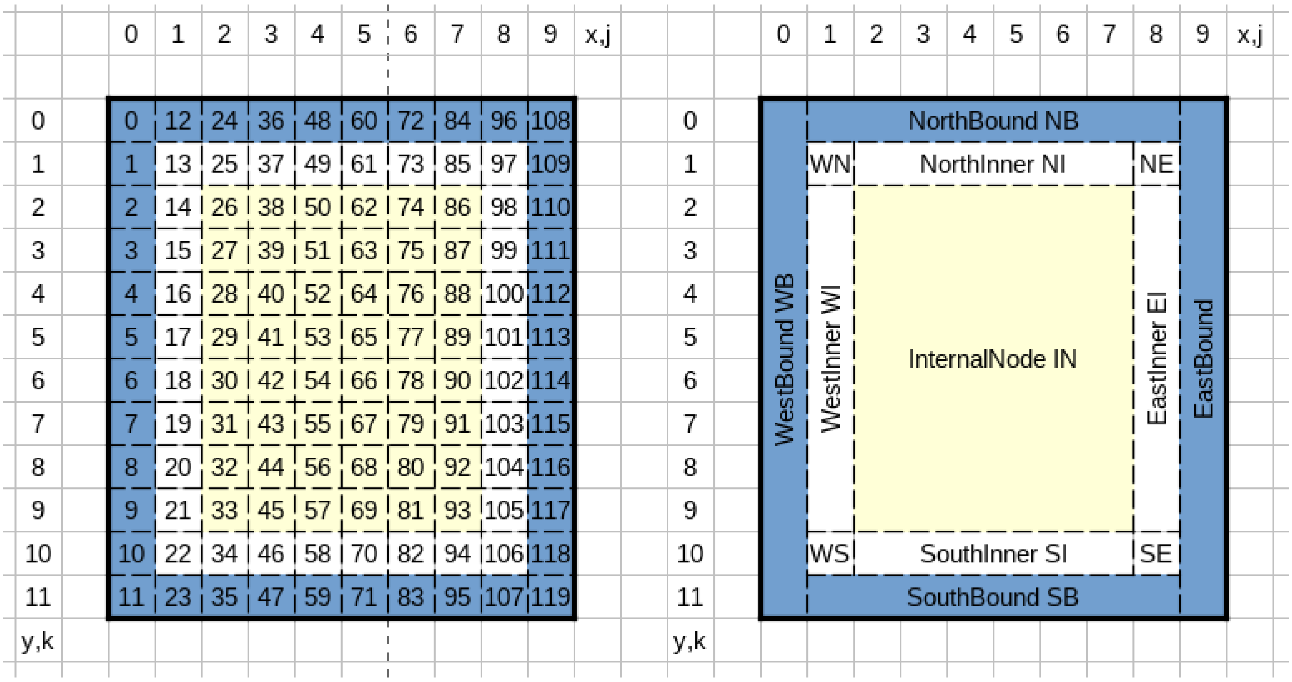


Figure 7: Index and System matrix areas

The system matrix is composed as follow :

$$M_{Sys} = \frac{1}{{\Delta_x}^2 {\Delta_y}^2}(D_x A_x + 2D_{xy} A_{xy} + D_y A_y) + M_{boundary}$$

Where $A_x$, $A_{xy}$, $A_y$ are composed according to the stencils of **figure 8**

For more versatility, $A_x$, $A_{xy}$ can be split in each in 2 matrix; one $A_{xi}$ for the internal nodes, one $A_{xb}$ for the inner bound (cells in white in the figures before the boundary blue cells). With this option the computation effort of the system matrix is lower if it has to be done several time in the same script (ie comparison if different boundary conditions).

# References

[1] H. Roknuzzaman, "Application of finite difference method for the analysis of a rectangular thin plate with eccentric opening," Feb. 2017. Available: https://www.researchgate.net/publication/313632391_Application_of_Finite_Difference_Method_for_the_Analysis_of_a_Rectangular_Thin_Plate_with_Eccentric_Opening

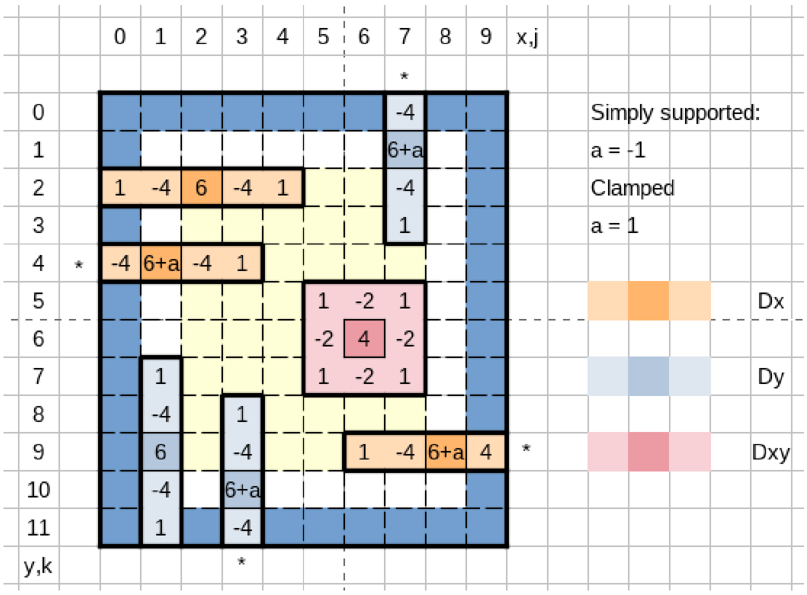


Figure 8: Dx, Dxy, Dy


[2] M. A. Zaman, "Numerical solution of the poisson equation using finite difference matrix operators," *Electronics*, vol. 11, no. 15, p. 2365, 2022, Available: https://www.mdpi.com/2079-9292/11/15/2365#B20-electronics-11-02365