Finite Difference Method applied to DML simulation. Part 2b rev2.

ChristianV (Homeswinghome@diyAudio)

06 Avr. 2025

Abstract

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

Contents

Disclaimer : this paper is written in the context of DIY DML building. 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.

Exported : April 6, 2025.

Revision 1. : The code is rewritten to be more compact and focused on the FDM operations.

Revision 2. : Base on the update of part1, 1b and 2, the free edge conditions are now included.

The pdf format of the paper is directly extracted from a python script See Github py2pdf for more information about this method.

1 Introduction

The part 2a is the application of part 1 and 1b in a Python script up to the calculation of the system matrix.

In this part 2b, the focus is on the modes (frequency and mode shape) and the comparison of the results of the script with some plate situations simulated with Elmer, Lisa or having exact solutions.

Elmer is a free multi-plateform FEM solver. More details about the simulation files used below.

The formulas to extend the simply supported plate case came first from [1] but were much too inaccurate. The Warburton approach [2] is now implemented.

Many thanks to Eric (Veleric@diyAudio) who helped by providing results from LISA. It was from a great help in bug fixing.

2 Eigenfrequencies

In part 1, the partial derevative equation (PDE) of the vibration of a plate was shown.

For an isotropic homogeneous material, the governing equation is :

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

The PDE in the case of an orthotropic material was also shown (refer to part 1).

The calculation of the eigenfrequencies is a specific case where this equation is solved.

Looking for the natural resonance frequencies, the external load is assumed to be null : $q(x,y,t) = 0$.

The second assumption is to separate the plate deviation w in a product of 2 functions :

$$w(x,y,t) = W(x,y)cos(\omega t)$$

w being in the time a cosine function, its second derivative over the time is also a cosine so the time “disappears” from the equation which becomes :

$$D(\frac{\partial^4 W}{\partial x^4} + 2\frac{\partial^4 W}{\partial x^2 \partial y^2} + \frac{\partial^4 W}{\partial y^4}) = \mu\omega^2 W \tag{2}$$

Which is with the finite difference method (see also part 1) FDM in the form :

$$M_{sys}W = \mu\omega^2 W \tag{3}$$

The modes are the eigenfrequencies of the M_{sys} and the mode shapes are the eigenvectors

This web page explains the concept of eigenvalue and of eigenvector; how numpy can extract them.

3 The testing plate

Different boundary conditions were tested and compare to FEM outputs.

The boundary conditions are denoted S (or SS) for simply supported, C for clamped or F for free for each edge and combined.

2 test cases denoted XXXX and OOOO were added where respectivly the 4 corners or the 4 central points of the edges have a null displacement condition.

See figure 1 for the plate representation in the FDM script.

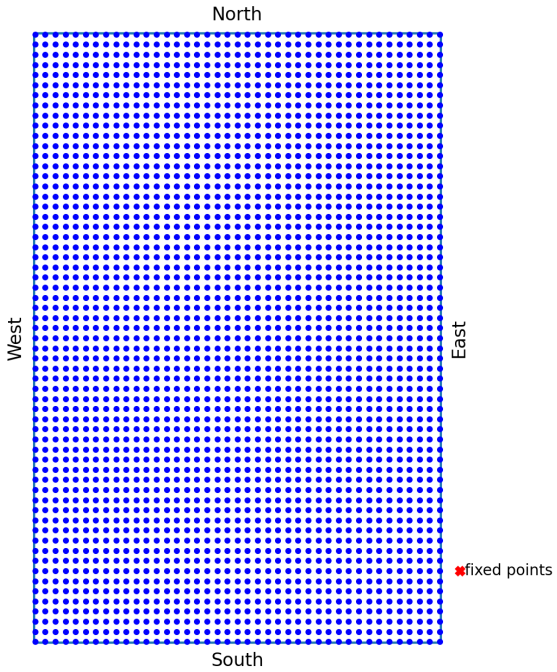

Figure 1: Testing plate

4 Computational efficiency

The time to get the result depends of course of the computer. For the one used for the first revisions, the results from Elmer were obtains much faster than with this FDM script which let suppose there are important possibilities of improvements but which need skills in coding.

By changing from an old I3 processor to a new (in 2023…) I7 (I7-1360P 13th generation), the response time is improved by a factor of about 20! With a 10mm grid and the 600x400mm test panel (2501 points), the modes are computed in few seconds. It is not an issue to get the results for more than 10 test cases. Note the performance change a lot with the power management options of the computer. Each case takes about 12s in energy saving mode on battery but only about 3s in performance mode on mains.

The execution time increases as the cube of the number of cells.

Possible heuristic : $t_{exec} = cte + N_{cells}^3/K$

See figure 2

5 FDM performances

5.1 First script with the specific SS and C boundary conditions implemented

With a 20mm grid, the precision for the 5 test cases are : -1.7, -.9, -1.2, -1.5, -1% over the 16th first modes.

With a 10mm grid, the precision for the 5 test cases are : .36, .21, .18, .36, .13% over the 16th first modes.

5.2 Free edge script

While the free edge conditions are implemented, the simply supported and clamped boundary conditions are replaced by zero displacement at the edge which give excellent results in the simply supported conditions and most probably acceptable for clamped condition. The error seen for clamped edges comes from plate dimension reduction linked to the double row of zero displacement points.

Because of the improvement in the computation efficiency, the 10mm grid was the only tested.

The script was tested over the 25 first modes of different combinaitions of edge conditions. The exact error values are in the script output section. Roughly, the mean error over the 25 first modes is below 1.5%. It

Cells	dx mm	mesh	i3-4030U	Model (I3)	I5-6200u	I7-1260P	I7-1360P
176	40	11x16	4,48	4,25			0,007
425	25	17x25	7,95	4,84			0,125
651	20	21x31	10,15	6,50	0,43	0,29	0,2
1734	12	34x51	46,7	47,65			1,4
2501	10	41x61	132,38	134,56	14,8	7,38	5,2
3876	8	51x76	495	489,46	48,16	26,86	24
9801	5				750,7	538,6	384

cte	4,2
K	1,20E+08

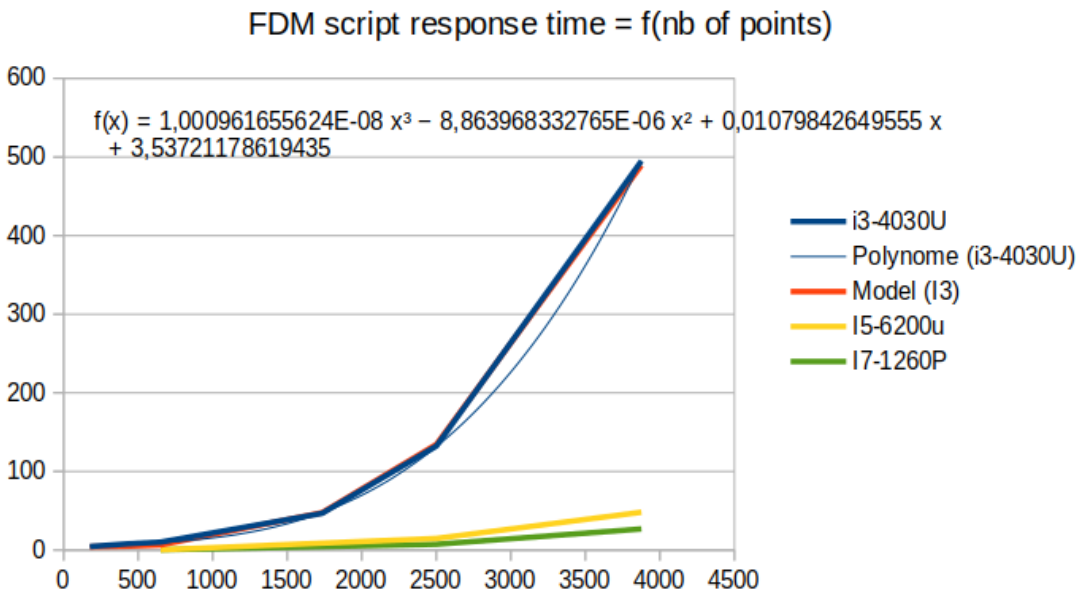

Figure 2: FDM performance

increases when 2 or more clamped edges are involved . The error with a clamped edge is a bit theoric related to the real panel dimensions, edge included or not.

To have a view beyond the 25 first modes, 250 modes were extracted from Elmer and the FDM and compared. The error seems to increase linearly with the frequency remaining below 5% up to 4kHz.This was not analyze to understand the role of the mesh size in the FDM and in Elmer.

See figure 3

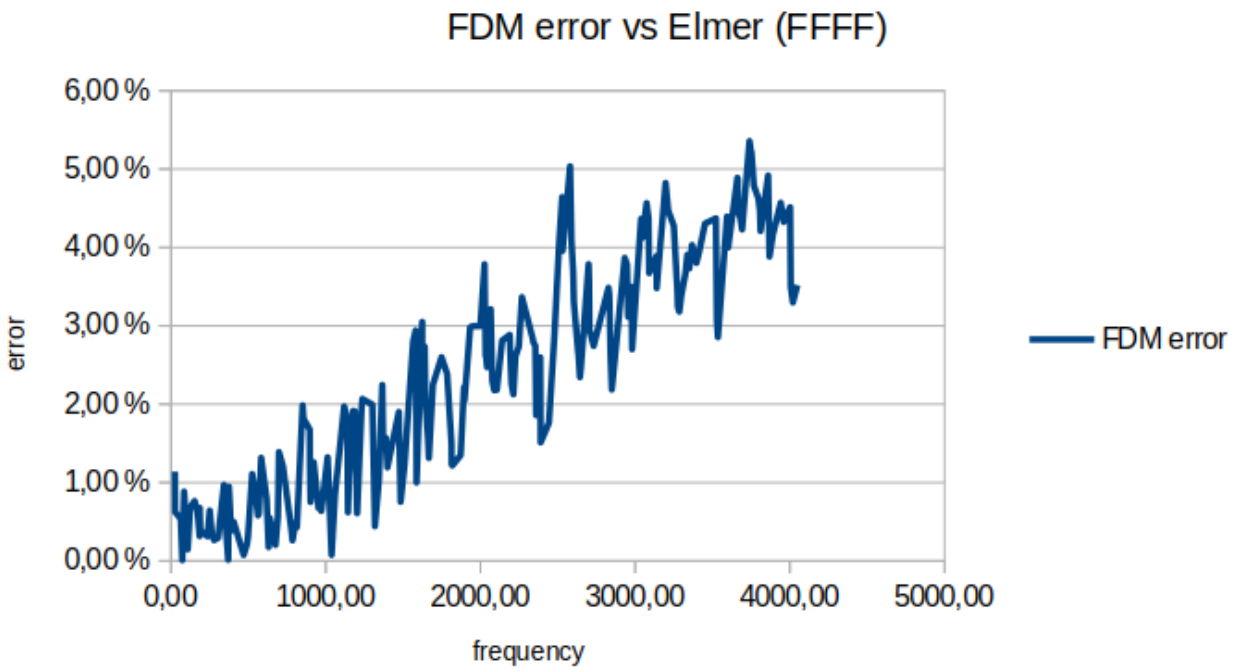

Figure 3: FDM performance

The formula tested from the Warburton's paper give very good results when simply supported and clamped edges are combined. Curiously, the results are not good when a free edge is involed while no bug was found…

6 Mode shapes

The FDM script extracts the mode shape. Below is an example presented in a colorized 2D. Other views are possible with matplotlib.

The mode shapes from the script were compare to the one from the FEM software LISA, at least for the first one of some configurations which was a good way to debug the script.

See figure 4

7 Conclusion on the FDM performances

The Python FDM script shows good enough performances in precision, computational time to be further used in DML design. Compared to other ways of simulation being very close to the reality in the description of the panel, it should offer an excellent flexibility to explore the different configuration of the DIYers' designs.

8 The script results

For each test case, a table is printed out with :

- FEM : eigenfrequencies in Hz from FEM tool (Elmer or Lisa)
- FDM : eigenfrequencies in Hz from the FDM script
- err% : FDM compared to FEM

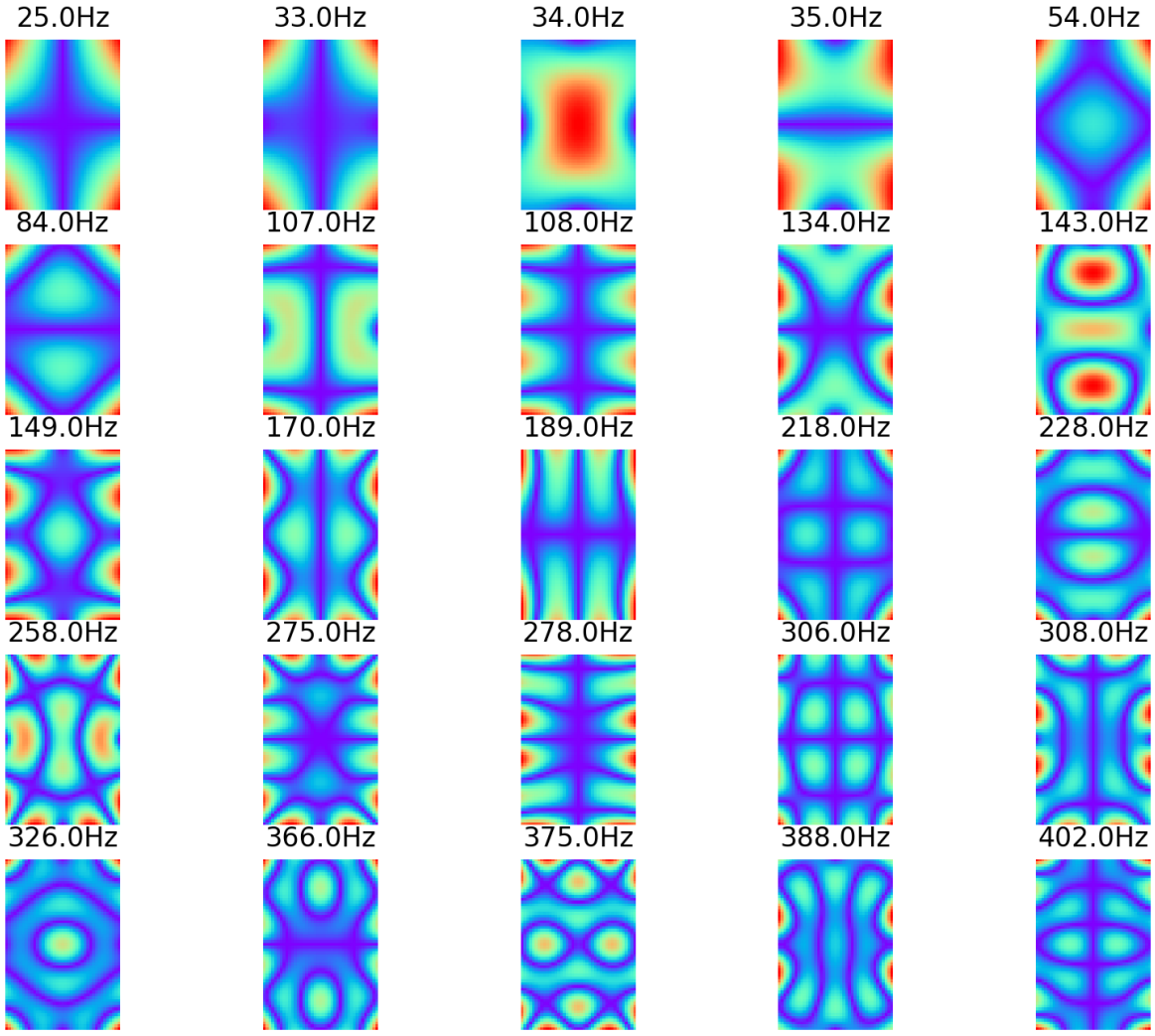

Figure 4: FDM performance

- Warb. : eigenfrequencies in Hz based on the formulas provided in the Warburton's paper
- err% : Warburton estimation compared to FEM

```
2025-04-06 13:52
The plate is Lx = 0.4 m by Ly = 0.6 m
The plate thickness is h = 0.015 m
The mesh is Nj = 41 points by Nk = 61 points
with a grid dx = 10.0 mm by dy = 10.0  mm
Young modulus E = 10 MPa, Density rho = 25.0 kg/m³
Bending stiffness Dx = Dy =  3.09 Nm, Areal density μ = 0.375 kg/m²
with Poisson's ratio nux = 0.3 and nuy = 0.3
Excecution time (system matrix preparation) = 0.1035 seconds for  2501  points
------------------------------------------------

Case # 0
Boundary conditions ['F [north]', 'F [west]', 'F [south]', 'F [east]']
Excecution time (solver) = 14.3606 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
        0         0         0       132         0
        0         0         0       196         0
        0         0         0       245         0
       25        25         0       277      1008
       27        27         0       319      1081
       58        58         0       380       555
       63        63         0       409       549
       72        73         1       411       470
       84        84         0       488       480
      107       108         0       505       371
      123       125         1       519       321
      151       151         0       584       286
      170       170         0       634       272
      183       184         0       650       255
      184       187         1       700       280
      188       189         0       711       278
      234       237         1       810       246
      249       249         0       837       236
      262       266         1       914       248
      276       278         0       930       236
      302       306         1       990       227
      339       339         0      1071       215
      349       351         0      1090       212
      365       370         1      1212       232
      371       371         0      1357       265
Mean error 0.51 %
Max error 1.6 %

Case # 1
Boundary conditions ['O [north]', 'O [west]', 'O [south]', 'O [east]']
Excecution time (solver) = 11.2198 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
       25        25         0         0      -100
       32        33         3         0      -100
       33        34         3         0      -100
       34        35         2         0      -100
       53        54         1         0      -100
       84        84         0         0      -100
      104       107         2         0      -100
      107       108         0         0      -100
      130       134         3         0      -100
      138       143         3         0      -100
      147       149         1         0      -100
```

```
      170       170         0         0      -100
      188       189         0         0      -100
      216       218         0         0      -100
      225       228         1         0      -100
      249       258         3         0      -100
      269       275         2         0      -100
      276       278         0         0      -100
      297       306         3         0      -100
      302       308         1         0      -100
      319       326         2         0      -100
      363       366         0         0      -100
      372       375         0         0      -100
      381       388         1         0      -100
      397       402         1         0      -100
Mean error 1.72 %
Max error 3.6 %

Case # 2
Boundary conditions ['F [north]', 'F [west]', 'S [south]', 'S [east]']
Excecution time (solver) = 10.6813 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
        6         6         0       101      1583
       27        27         0       155       474
       47        47         0       197       319
       69        69         0       228       230
       76        76         0       258       239
      124       125         0       323       160
      137       137         0       338       146
      145       145         0       348       140
      173       173         0       411       137
      194       195         0       439       126
      225       226         0       442        96
      231       231         0       496       114
      287       288         0       557        94
      298       297         0       562        88
      301       302         0       602       100
      326       325         0       620        90
      349       347         0       706       102
      379       379         0       729        92
      396       399         0       815       105
      406       406         0       823       102
      455       457         0       885        94
      491       487         0       946        92
      509       504         0       972        90
      517       518         0      1083       109
      536       532         0      1216       126
Mean error 0.0 %
Max error 0.8 %

Case # 3
Boundary conditions ['S [north]', 'F [west]', 'S [south]', 'F [east]']
Excecution time (solver) = 10.1673 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
       12        12         0        83       591
       26        27         3       132       407
       49        49         0       193       293
       68        69         1       203       198
       82        83         1       243       196
      110       110         0       296       169
      129       131         1       318       146
      131       133         1       361       175
```

```
      192       192         0       409       113
      196       197         0       413       110
      197       201         2       417       111
      218       220         0       485       122
      238       241         1       538       126
      287       292         1       585       103
      306       308         0       586        91
      309       315         1       632       104
      327       331         1       707       116
      357       358         0       710        98
      399       405         1       808       102
      401       406         1       867       116
      402       411         2       912       126
      439       443         0       933       112
      460       466         1       985       114
      471       480         1      1086       130
      516       529         2      1212       134
Mean error 1.24 %
Max error 3.8 %

Case # 4
Boundary conditions ['F [north]', 'S [west]', 'F [south]', 'S [east]']
Excecution time (solver) = 10.7177 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
       27        27         0        65       140
       36        37         2       115       219
       64        65         1       156       143
      111       111         0       189        70
      114       114         0       211        85
      121       121         0       286       136
      153       154         0       288        88
      187       188         0       299        59
      206       208         0       358        73
      251       250         0       385        53
      261       260         0       412        57
      281       284         1       437        55
      286       286         0       498        74
      295       295         0       509        72
      350       352         0       538        53
      381       382         0       558        46
      411       408         0       640        55
      428       431         0       663        54
      448       444         0       744        66
      457       454         0       752        64
      491       490         0       814        65
      504       504         0       871        72
      529       532         0       898        69
      548       548         0      1004        83
      560       555         0      1133       102
Mean error 0.21 %
Max error 2.7 %

Case # 5
Boundary conditions ['S [north]', 'F [west]', 'F [south]', 'F [east]']
Excecution time (solver) = 10.5594 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
        0         0         0       120         0
       12        12         0       176      1366
       18        18         0       232      1188
       42        43         2       251       497
       60        60         0       295       391
```

```
       69        69         0       347       402
       88        89         1       378       329
      102       102         0       398       290
      130       130         0       463       256
      155       156         0       467       201
      157       158         0       482       207
      179       178         0       550       207
      211       211         0       607       187
      223       222         0       621       178
      232       234         0       659       184
      249       249         0       685       175
      270       272         0       775       187
      327       328         0       790       141
      339       337         0       887       161
      349       346         0       901       158
      350       352         0       965       175
      367       366         0      1021       178
      377       376         0      1054       179
      434       435         0      1168       169
      449       449         0      1305       190
Mean error 0.17 %
Max error 2.29 %

Case # 6
Boundary conditions ['F [north]', 'S [west]', 'F [south]', 'F [east]']
Excecution time (solver) = 10.4949 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
        0         0         0       113         0
       12        12         0       174      1350
       36        36         0       210       483
       44        44         0       254       477
       57        57         0       280       391
       86        86         0       355       312
       93        94         1       361       288
      140       140         0       367       162
      146       147         0       435       197
      155       155         0       474       205
      160       160         0       480       200
      193       194         0       528       173
      223       224         0       570       155
      252       253         0       603       139
      260       259         0       640       146
      295       293         0       644       118
      306       305         0       740       141
      322       323         0       774       140
      334       336         0       836       150
      346       346         0       856       147
      383       381         0       910       137
      409       410         0       995       143
      437       439         0      1006       130
      445       444         0      1125       152
      494       496         0      1267       156
Mean error 0.1 %
Max error 1.0 %

Case # 7
Boundary conditions ['X [north]', 'X [west]', 'X [south]', 'X [east]']
Excecution time (solver) = 11.2234 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
       11        11         0         0      -100
       26        27         3         0      -100
```

```
       32        32         0         0      -100
       42        42         0         0      -100
       65        66         1         0      -100
       72        73         1         0      -100
       87        88         1         0      -100
      110       113         2         0      -100
      138       139         0         0      -100
      139       142         2         0      -100
      144       146         1         0      -100
      170       170         0         0      -100
      206       211         2         0      -100
      209       212         1         0      -100
      223       226         1         0      -100
      226       230         1         0      -100
      280       285         1         0      -100
      283       287         1         0      -100
      298       303         1         0      -100
      307       316         2         0      -100
      350       351         0         0      -100
      350       354         1         0      -100
      377       379         0         0      -100
      386       391         1         0      -100
      398       412         3         0      -100
Mean error 1.41 %
Max error 3.8 %

Case # 8
Boundary conditions ['S [north]', 'S [west]', 'S [south]', 'S [east]']
Excecution time (solver) = 9.0550 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
       40        40         0        40         0
       77        78         1        78         1
      124       125         0       125         0
      140       140         0       140         0
      161       162         0       162         0
      223       225         0       225         0
      227       227         0       228         0
      265       265         0       266         0
      302       302         0       303         0
      310       312         0       313         0
      340       339         0       341         0
      363       364         0       366         0
      423       423         0       425         0
      449       452         0       454         1
      463       459         0       463         0
      477       475         0       501         5
      499       497         0       563        12
      560       559         0       566         1
      560       559         0       651        16
      561       563         0       717        27
      640       635         0       754        17
      645       646         0       764        18
      699       699         0       817        16
      717       708        -1       905        26
      723       719         0      1017        40
Mean error 0.03 %
Max error 1.2 %

Case # 9
Boundary conditions ['C [north]', 'C [west]', 'C [south]', 'C [east]']
Excecution time (solver) = 9.6949 seconds for  2501  points
```

```
      FEM       FDM      err%     Warb.      err%
       76        80         5        77         1
      118       124         5       119         0
      186       197         5       189         1
      188       198         5       190         1
      225       238         5       229         1
      285       297         4       289         1
      290       307         5       295         1
      353       375         6       358         1
      383       404         5       390         1
      389       414         6       397         2
      407       425         4       413         1
      452       481         6       462         2
      502       528         5       512         1
      542       575         6       553         2
      554       577         4       584         5
      573       609         6       623         8
      609       648         6       672        10
      647       679         4       687         6
      658       697         5       776        17
      670       715         6       866        29
      725       754         4       893        23
      757       808         6       906        19
      799       845         5       969        21
      816       854         4      1058        29
      848       900         6      1172        38
Mean error 5.53 %
Max error 6.7 %

Case # 10
Boundary conditions ['S [north]', 'S [west]', 'C [south]', 'C [east]']
Excecution time (solver) = 16.7274 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
       56        58         3        62        10
       96        99         3       105         9
      154       158         2       160         3
      163       167         2       173         6
      191       197         3       204         6
      255       261         2       267         4
      255       263         3       272         6
      308       316         2       315         2
      344       355         3       358         4
      345       355         2       365         5
      373       380         1       386         3
      406       420         3       426         4
      461       473         2       484         4
      494       510         3       520         5
      515       524         1       527         2
      517       530         2       570        10
      553       569         2       638        15
      603       616         2       638         5
      608       627         3       731        20
      614       633         3       795        29
      682       693         1       838        22
      700       723         3       849        21
      748       769         2       905        20
      769       784         1       998        29
      781       799         2      1116        42
Mean error 2.65 %
Max error 3.5 %
```

```
Case # 11
Boundary conditions ['S [north]', 'C [west]', 'S [south]', 'C [east]']
Excecution time (solver) = 9.5196 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
       71        74         4        71         0
       99       103         4       100         1
      155       158         1       157         1
      183       194         6       185         1
      213       224         5       216         1
      239       241         0       241         0
      266       278         4       270         1
      346       350         1       351         1
      349       358         2       352         0
      350       372         6       355         1
      381       404         6       387         1
      434       458         5       442         1
      452       463         2       459         1
      485       485         0       522         7
      512       537         4       581        13
      571       595         4       614         7
      585       607         3       627         7
      603       640         6       670        11
      617       641         3       749        21
      646       643         0       854        32
      656       696         6       863        31
      734       751         2       897        22
      744       770         3       953        28
      747       775         3      1034        38
      832       825         0      1138        36
Mean error 3.48 %
Max error 6.2 %

Case # 12
Boundary conditions ['S [north]', 'S [west]', 'C [south]', 'S [east]']
Excecution time (solver) = 9.0735 seconds for  2501  points
      FEM       FDM      err%     Warb.      err%
       44        44         0        46         4
       88        89         1        90         2
      126       127         0       131         3
      157       160         1       159         1
      168       170         1       174         3
      236       240         1       243         2
      250       255         2       254         1
      267       266         0       271         1
      307       308         0       315         2
      330       336         1       337         2
      369       375         1       373         1
      373       377         1       384         2
      448       456         1       456         1
      464       461         0       469         1
      465       472         1       477         2
      503       502         0       512         1
      512       520         1       580        13
      567       570         0       597         5
      583       592         1       674        15
      591       601         1       722        22
      658       664         0       766        16
      679       689         1       793        16
      718       709        -1       834        16
      726       737         1       928        27
      756       749         0      1046        38
```

```
Mean error 0.88 %
Max error 2.0 %
```

9 Elmer

9.1 Geometry (rectangular_plate.geo)

```
// Inputs
Lx = 0.4; //m
Ly = 0.6; //m
gridsize = Lx / 20;

// All numbering counterclockwise from bottom-left corner
Point(1) = {-Lx/2, -Ly/2, 0, gridsize};
Point(2) = {Lx/2, -Ly/2, 0, gridsize};
Point(3) = {Lx/2, Ly/2, 0, gridsize};
Point(4) = {-Lx/2, Ly/2, 0, gridsize};
Line(1) = {1, 2};               // bottom line
Line(2) = {2, 3};               // right line
Line(3) = {3, 4};               // top line
Line(4) = {4, 1};               // left line
Line Loop(5) = {1, 2, 3, 4};
// the order of lines in Line Loop is used again in surfaceVector[]
Plane Surface(6) = {5};
```

The rectangular_plate.msh is obtain from the gmsh application.

9.2 Mesh

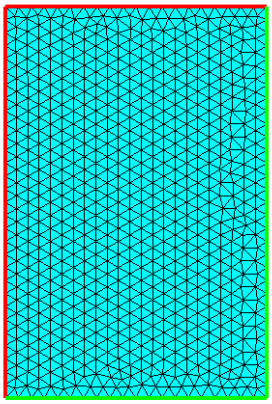

Figure 5: plate mesh

See figure 5

9.3 Solver (case.sif)

```
Header
  CHECK KEYWORDS Warn
  Mesh DB "." "."
  Include Path ""
  Results Directory ""
End

Simulation
  Max Output Level = 5
```

```
  Coordinate System = Cartesian
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = Steady state
  Steady State Max Iterations = 1
  Output Intervals = 1
  Timestepping Method = BDF
  BDF Order = 1
  Solver Input File = case.sif
  Post File = case.vtu
End

Constants
  Gravity(4) = 0 -1 0 9.82
  Stefan Boltzmann = 5.67e-08
  Permittivity of Vacuum = 8.8542e-12
  Boltzmann Constant = 1.3807e-23
  Unit Charge = 1.602e-19
End

Body 1
  Target Bodies(1) = 1
  Name = "Body 1"
  Equation = 1
  Material = 1
End

Solver 1
  Equation = Elastic Plates
  Variable = -dofs 3 Deflection
  Eigen System Values = 25
  Eigen Analysis = True
  Eigen System Select = Smallest magnitude
  Procedure = "Smitc" "SmitcSolver"
  Exec Solver = Always
  Stabilize = True
  Bubbles = False
  Lumped Mass Matrix = False
  Optimize Bandwidth = True
  Steady State Convergence Tolerance = 1.0e-5
  Nonlinear System Convergence Tolerance = 1.0e-7
  Nonlinear System Max Iterations = 1
  Nonlinear System Newton After Iterations = 3
  Nonlinear System Newton After Tolerance = 1.0e-3
  Nonlinear System Relaxation Factor = 1
  Linear System Solver = Direct
  Linear System Direct Method = Umfpack
End

Equation 1
  Name = "Plate Equation"
  Active Solvers(1) = 1
End

Material 1
  Name = "EPS"
  Porosity Model = Always saturated
  Tension = 0.0
  Youngs modulus = 10e6
  Thickness = 0.015
  Poisson ratio = 0.
  Density = 25
```

```
End

Boundary Condition 1
  Target Boundaries(2) = 3 4
  Name = "fixed"
  Deflection 2 = 0
  Deflection 3 = 0
  Deflection 1 = 0
End

Boundary Condition 2
  Target Boundaries(2) = 1 2
  Name = "Simply supported"
  Deflection 1 = 0
End
```

The Elmer results are stored in the elmer() function to be displayed along with the script results.

10 Python script code

10.1 Import modules

```
import os  # for py2pdf
import shutil  # for py2pdf backup
import time  # for script time performance
import christianpylib.py2pdf as p2p  # import py2pdf library
import numpy as np
# from fdm_applied_to_dml_part2b1 import Bound_cond
import mod_fdm_print_FFFF as fdmp  # import the module of specific functions to print and plot
import mod_fdm_FFFF as fdmbm  # import the module of specific functions to build the matrices of a free
import mod_fdm_elmer as fdmelm
import matplotlib.pyplot as plt
from scipy import linalg
```

10.2 Functions

```
Empty section : all the functions are in external modules
```

10.3 Prepare py2pdf

```
# lines to be included for py2pdf export
script_name = os.path.basename(__file__).split(".")[0]  # get this script file name without extension
py2pdfdir = "../data/part2b2" # where the script logs are saved
logfile = py2pdfdir + "/DML_FDM3b.txt"  # define the logfile
p2p.clearlog(logfile)  # clear the logfile (in case script is ran several times)
logfile2 = py2pdfdir + "/DML_FDM3b_2.txt"  # define the logfile to store eigenfrequencies of FFFF
p2p.clearlog(logfile2)  # clear the logfile (in case script is ran several times)
```

10.4 Parameters

```
# Test cases reported in this paper
# Boundary conditions C = clamped, SS = simply supported, F = free, North, West, South, East
BoundDict1 = {
    "North": "F",
    "West": "F",
    "South": "F",
    "East": "F"
}
BoundDict2 = {
```

```
    "North": "O",
    "West": "O",
    "South": "O",
    "East": "O"
}
BoundDict3 = {
    "North": "F",
    "West": "F",
    "South": "SS",
    "East": "SS"
}
BoundDict4 = {
    "North": "SS",
    "West": "F",
    "South": "SS",
    "East": "F"
}
BoundDict5 = {
    "North": "F",
    "West": "SS",
    "South": "F",
    "East": "SS"
}
BoundDict6 = {
    "North": "SS",
    "West": "F",
    "South": "F",
    "East": "F"
}
BoundDict7 = {
    "North": "F",
    "West": "SS",
    "South": "F",
    "East": "F"
}
BoundDict8 = {
    "North": "X",
    "West": "X",
    "South": "X",
    "East": "X"
}
BoundDict9 = {
    "North": "SS",
    "West": "SS",
    "South": "SS",
    "East": "SS"
}
BoundDict10 = {
    "North": "C",
    "West": "C",
    "South": "C",
    "East": "C"
}
BoundDict11 = {
    "North": "SS",
    "West": "SS",
    "South": "C",
    "East": "C"
}
BoundDict12 = {
    "North": "SS",
```

```
    "West": "C",
    "South": "SS",
    "East": "C"
}
BoundDict13 = {
    "North": "SS",
    "West": "SS",
    "South": "C",
    "East": "SS"
}
# New way to define boundary condition since part2a rev2
Bound_case2 = [BoundDict1, BoundDict2, BoundDict3, BoundDict4, BoundDict5, BoundDict6,
    BoundDict7, BoundDict8, BoundDict9, BoundDict10, BoundDict11, BoundDict12, BoundDict13]
# For compatibility with previous version
Direction = ["north", "west", "south", "east"]

case_number = 13

Bound_case = [
    ["F", "F", "F", "F"],
    ["O", "O", "O", "O"],
    ["F", "F", "S", "S"],
    ["S", "F", "S", "F"],
    ["F", "S", "F", "S"],
    ["S", "F", "F", "F"],
    ["F", "S", "F", "F"],
    ["X", "X", "X", "X"],
    ["S", "S", "S", "S"],
    ["C", "C", "C", "C"],
    ["S", "S", "C", "C"],
    ["S", "C", "S", "C"],
    ["S", "S", "C", "S"],
]


case2print = 1  # This test case will be plotted

# Modes to be displayed
m = 5
n = m  # m by n modes are displayed

# Dimensions (in m otherwise mentionned)
# do not change the values (used in Elmer for comparison)
Lxmm = 400
Lymm = 600  # use locally mm to avoid rounding problems
h = 0.015  # plate thickness

# Material
# do not change the values (used in Elmer for comparison)
Ex = 10e6
Ey = 10e6  # Young modulus x and y direction in Pa
rho = 25.0  # density in kg/m^3
nux = 0.3  # Poisson ration
nuy = nux

Dx = Ex * h**3 / 12 / (1 - nux**2)  # bending stiffness
Dy = Ey * h**3 / 12 / (1 - nuy**2)  # bending stiffness
mu = rho * h  # areal density

# Mesh
meshmm = 10.1  #mm, dxmm and dymm will adjusted just less than mesh for Nj and Nk odd
```

```
# dxmm = 10
# dymm = dxmm

# Some parameters
nj = int(Lxmm/meshmm/2) + 1
Nj= 2*nj+1
nk = int(Lymm/meshmm/2) + 1
Nk= 2*nk+1
# Nj = Lxmm // dxmm + 1  # number of points in the mesh, j index = x direction = horizontal sides
# Nk = Lymm // dymm + 1  # same in k index = y direction = vertical sides
# some conversion to meter
Lx = Lxmm / 1000
Ly = Lymm / 1000
# dx = dxmm / 1000
# dy = dymm / 1000
dx = Lx/(Nj-1)
dy = Ly/(Nk-1)
dxmm = int(100000 * dx)/100 # in mm and rounded. Do not use for calculation
dymm = int(100000 * dy)/100 # in mm and rounded. Do not use for calculation
# 2D matrix containing the index i of each points
plate2d = np.transpose(np.arange(Nk * Nj).reshape(Nj, Nk))
# Prepare the sub-areas
# AreasLim, AreasName, AreasMatrix are dictionaries
# key is the area short name (ie Sx, EW, IE...)
AreasLim, AreasName = fdmbm.AreasDict()
# AreasMatrix = fdmbm.AreasMat(AreasLim)
AreasSubMat = fdmbm.AreasMat(AreasLim, plate2d)

# Summary
p2p.print_twice(logfile, "The plate is Lx =", Lx, "m by Ly =", Ly, "m")
p2p.print_twice(logfile, "The plate thickness is h =", h, "m")
p2p.print_twice(logfile, "The mesh is Nj =", Nj, "points by Nk =", Nk, "points")
p2p.print_twice(logfile, "with a grid dx =", dxmm, "mm by dy =", dymm, " mm")
p2p.print_twice(
    logfile, "Young modulus E =", int(Ex / 1e6), "MPa, Density rho =", rho, "kg/m³")
p2p.print_twice(
    logfile,
    "Bending stiffness Dx = Dy = ",
    int(Dx * 100) / 100, "Nm, Areal density μ =", mu, "kg/m²")
p2p.print_twice(logfile, "with Poisson's ratio nux =", nux, "and nuy =", nuy)
```

10.5 System matrix filling process

```
tic = time.perf_counter()  # optional, start of a timer for time performance check
'''
# Fill the cell type matrix (inner, boundary...)
plate2d = np.zeros([Nk, Nj])
plate2d = fdmp.fillcelltype(plate2d)

Imat = np.transpose(np.arange(Nk * Nj).reshape(Nj, Nk))
'''
# Create the system matrix
SysMat = np.zeros([Nk * Nj, Nk * Nj], dtype=float)  # System Matrix
Sysmatcase = np.zeros([Nk * Nj, Nk * Nj], dtype=float)  # System Matrix

# Let's fill the system submatrices
Ax, Ay, Axy = fdmbm.fill_main_matrices(Nj, Nk, nux, nuy, AreasSubMat)
# Magic : here is the System Matrix
# --------------------------------
SysMat = (Ax / dx**4 + Ay / dy**4 + 2*Axy / dx**2 / dy**2)  # FFFF version
```

```
''' # For debug, to see the system matrix
N = 50  # number of elements to print
fdmp.report_mat(logfile, SysMat, plate2d, "SysMat raw", N)
'''
SysMat = (Dx * SysMat) / mu
# ------------

toc = time.perf_counter()  # to evaluate time performance of the algorithm
p2p.print_twice(logfile, f"Excecution time (system matrix preparation) = {toc - tic:0.4f} seconds for "
p2p.print_twice(logfile, "-----------------------------------------------")
p2p.print_twice(logfile, "")
```

10.6 Solver

```
for Case in range(case_number):
    BoundDict = Bound_case2[Case]
    Bound_cond = Bound_case[Case]  # For compatibility with previous version (before adding free edge)

    # Fixed points ------------------------------------
    # Apply Boundary Conditions
    if Bound_cond == ["X", "X", "X", "X"]:
        FixedPoints = np.array([0, Nk-1, Nk*(Nj-1), Nk*Nj -1])
    elif Bound_cond == ["O", "O", "O", "O"]:
        # FixedPoints = np.array([0, (Nk+1)/2, Nk*(Nj+1)/2, Nk*Nj -1 - (Nk+1)/2])
        FixedPoints = np.array([int((Nk-1)/2), int(Nk*(Nj-1)/2), int((Nj+1)/2*Nk-1), int(Nj*Nk-(Nk+1)/2
    else:
        FixedPoints = fdmbm.Fixedpoints(BoundDict, AreasSubMat)
    # Apply fixed points
    Sysmatcase = fdmbm.applyfixedpoints(SysMat, FixedPoints)
    # Specific to FFFF, null mean displacement added
    # --------------------------------

    BC = fdmbm.boundcoeff(Bound_cond)
    BCname = ''.join(Bound_cond) # used to adress the test case in elmer() function
    plotboundcond = ["", "", "", ""]
    for i in range(4):
        plotboundcond[i] = Bound_cond[i] + " [" + Direction[i] + "]"

    p2p.print_twice(logfile, "Case #", Case)
    p2p.print_twice(logfile, "Boundary conditions", plotboundcond)

    # p2p.print_twice(logfile, "Boundary coeff. North=", BC[0], " West=", BC[1], " South=", BC[2], " Ea

    # Here is the solver (eigenfreq)
    tic = time.perf_counter()  # optional, start of a timer for time performance check

    # Compute the eigenfrequencies and the mode shapes
    vals, vecs = linalg.eig(Sysmatcase) # vals is 1D NjNy points (complex values, 1+1j for the fixed po
    # Clean the results by removing 1 values and complex from the eigenvalues
    vals = vals[vals != 1.0 + 0j]  # remove the value 1.+0j
    # print("vals (no 1):", vals.shape)
    vals = np.real(vals)

    modefreq = np.zeros([len(vals), 2])  # prepare an array for the eigenfrequencies
    modefreq[:, 1] = np.abs(vals)**0.5 / 2 / np.pi  # extract the frequencies
    modefreq[:, 0] = np.arange(0, len(vals), 1)  # give an index
    modefreq = modefreq[modefreq[:, 1].argsort()]  # sort
    # for ii in range(50):
    #     print(modefreq[ii, 0], modefreq[ii, 1])
```

```
modefreq = modefreq.astype(int)  # keep interger format to reduce the number of digit
# now modefreq is filled : modefreq[i,0] = mode index, modefreq[i,1] = mode frequency
toc = time.perf_counter()  # to evaluate time performance of the algorithm
p2p.print_twice(logfile, f"Excecution time (solver) = {toc - tic:0.4f} seconds for ", Nj * Nk, " po

# Show the plate
if Case == 0:
    fdmp.plot_plate2(Nj, Nk, FixedPoints, py2pdfdir + "/plate.png")  # show the mesh
```

10.7 Script output printing

```
results = np.zeros([m * n, 8])  # prepare array result
# results table columns
# | 0  |  1  |    2     |    3    |  4   |     5       |   6   |         7       |
# | mx |  ny |elmer freq|mode freq| error |theoric freq | error |mode shape index |
results[:, 3] = modefreq[: m * n, 1]  # eigenfrequency in col 3
results[:, 7] = modefreq[: m * n, 0]  # mode shape number in col 7

for i in range(m * n):
    # results[i,6] = np.mean(vecs[:, int(results[i,7])]) # mode shape mean value
    results[i, 2] = fdmelm.elmer(BCname, i, m * n - 1)  # elmer result in col 2

drivingpoint = np.zeros([Nk, Nj])

if Case == case2print:
    # if Case >= 0: # plot all the cases
    fig = plt.figure(figsize=(10, 8))  # create the figure

for i in range(m * n):
    modeshape = np.real(np.transpose(vecs[:, int(results[i, 7])].reshape((Nj, Nk))))
    # modeshape is the 2D matrix showing the plate deviation in the plate coordinates
    ind = np.unravel_index(
        np.argmax(np.absolute(modeshape), axis=None), modeshape.shape
    )
    # ind returns the coordinates of the maximum sees in modeshape
    diry = modeshape[:, ind[1]]
    # diry is the 1D slice of modeshape in y direction at the maximum
    dirx = modeshape[ind[0], :]
    # dirx is the 1D slice of modeshape in x direction at the maximum
    # the mode numbers are obtains by checking how lany times dirx, diry signs change
    mx = fdmelm.signchange(dirx)
    ny = fdmelm.signchange(diry)
    fdmresult = results[i, 3]
    results[i, 0] = mx
    results[i, 1] = ny
    # results[i, 5] = fdmelm.modes(Bound_cond, Lx, Ly, Dx, mu, mx, ny)  # formula result in 5
    # results[i, 5] = fdmelm.warburton(Bound_cond, Lx, Ly, Dx, nux, mu, mx, ny)
    '''
    if results[i, 7] != 0:  # test of driving point similar to Zenker's paper
        mode = np.absolute(modeshape)
        drivingpoint = drivingpoint + (mode / np.max(mode)) ** 0.3
        if i > 0:
            drivingpoint = drivingpoint / np.max(drivingpoint)
    '''

    # if Case == case2print:
    if Case >= 0: # plot all the cases
        plt.subplot(m, n, i + 1)
        # if Case == 8: # FFFF in dirty mode
            # mode2print = np.absolute(modeshape)
```

```
            # plt.imshow(np.absolute(mode2print - np.amax(mode2print)), cmap = 'rainbow')
            # else:
        plt.imshow(np.absolute(modeshape), cmap = 'rainbow')
        # plt.plot(ind[1], ind[0], "r*")
        # plt.title(str(int(results[i,0]))+","+str(int(results[i,1])))
        # plt.title(str(mx) + "," + str(ny))
        plt.title(str(fdmresult)+"Hz")
        plt.axis("equal")
        plt.axis("off")

mode_estimation = np.zeros([m * n, 1])
for i in range(m * n):
    mx = i//m + 1
    ny = i%m + 1
    mode_estimation[i] = fdmelm.warburton(Bound_cond, Lx, Ly, Dx, nux, mu, mx, ny)
results[:, 5] = np.sort(mode_estimation, axis=None)

# results[:,6] = 100*results[:,6]/np.max(abs(results[:,6]))
# results[:,6] = results[:,6].astype(int)
for i in range(m * n):
    if results[i, 2] != 0:
        error = 1000 * (results[i, 3].astype(int) / results[i, 2].astype(int) - 1)
        results[i, 4] = error.astype(int) / 10
    else:
        results[i, 4] = 0
mean_error = int(100 * np.average(results[:, 4])) / 100
max_error = int(100 * np.amax(results[:,4])) / 100
results[:, 4] = results[:, 4].astype(int)

for i in range(m * n):
    if results[i, 2] != 0:
        error = 1000 * (results[i, 5].astype(int) / results[i, 2].astype(int) - 1)
        results[i, 6] = error.astype(int) / 10
    else:
        results[i, 6] = 0
results[:, 6] = results[:, 6].astype(int)

'''
# save the eigenfrequencies for comparison with Elmer (above the 25 first one)
for i in range(1000):
    p2p.print_twice(logfile2, modefreq[i, 1])
'''
# print(results)
spacing = 10
legend = ["m", "n", "FEM", "FDM", "err%", "Warb.", "err%"]
ac = ""
for j in range(2, 7): # m, n no more printed out (no meaning with free edge)
    ab = legend[j]
    ab = (spacing - len(ab) - 1) * " " + ab + " "
    ac = ac + ab
p2p.print_twice(logfile, ac)
for i in range(m * n):
    ac = ""
    for j in range(2, 7):
        ab = str(int(results[i, j]))
        ab = (spacing - len(ab) - 1) * " " + ab + " "
        ac = ac + ab
    p2p.print_twice(logfile, ac)
p2p.print_twice(logfile, "Mean error", mean_error, "%")
p2p.print_twice(logfile, "Max error", max_error, "%")
```

```
    if Case == case2print:
        # if Case >= 0: # plot all the cases
        plt.suptitle(
            "Mode shapes "
            + Bound_cond[0]
            + Bound_cond[1]
            + Bound_cond[2]
            + Bound_cond[3]
        )
        plt.show()
        fig.savefig(
            py2pdfdir + "/modeshape.png", bbox_inches="tight", dpi=200
        )  # this is the key line to store the matplotlib output

    p2p.print_twice(logfile, "")
```

10.8 py2pdf

```
print("start py2pdf")

# WARNING : no output after this line can be recorded automatically (reason is the markdown is created
# export the markdown from the python file
msg, date_ext = p2p.py2md(script_name, "# ~~")
# prepare the command to launch the pdf report creation
markdownfile = "../docs/fdm_applied_to_dml_part2b2.md"
pdffile = "../docs/FDM_applied_to_DML_part2b2.pdf"
pdfbackup = "../p2pbackup/FDM_applied_to_DML_part2b2" + date_ext + ".pdf.bak"

cmd = ('pandoc --pdf-engine=xelatex --variable mainfont="Arial" '+ markdownfile + ' -s --citeproc --num

# Run the command and capture the return code
return_code = os.system(cmd)
# Check the return code to determine success or failure
if return_code == 0:
    print("Pandoc command executed successfully. Hurrah the pdf was created!")
    shutil.copy(pdffile, pdfbackup)
else:
    print("Error: Pandoc command failed with exit code.", return_code)
```

11 References

[1] R. Maso, “Application of boundary edge effect corrections to the vibration of beams and rectangular plates.” PhD thesis, Carleton University, 1993. Available: https://curve.carleton.ca/system/files/etd/10817845-5151-47ac-9066-6fc2e360a6f2/etd_pdf/832572c0d80b3466b8f401a713c81266/maso-applicationofboundaryedgeeffectcorrections.pdf

[2] G. Warburton, “The vibration of rectangular plates,” *Proceedings of the Institution of Mechanical Engineers*, vol. 168, no. 1, pp. 371–384, 1954.

[3] A. W. Leissa, *Vibration of plates*, vol. 160. Scientific and Technical Information Division, National Aeronautics and …, 1969. Available: https://ntrs.nasa.gov/api/citations/19700009156/downloads/19700009156.pdf