Title
Geometric Classifications of Free-Form Surfaces in the Imaginary Parameter Space of DFT-Based Technique

Author(s)
SAWADA Kiichiro / Nakano Sousuke

Journal
Journal of the International Association for Shell and Spatial Structures 66 (4) , 272 - 283

Published
2025-12-01

URL (The Version of Record)
https://doi.org/10.20898/j.iass.2025.013

Authors Accepted Manuscript for Sawada, K., & Nakano, S. (2025). J. of the IASS, 66(4), 272-283.

GEOMETRIC CLASSIFICATIONS OF FREE-FORM SURFACES IN THE IMAGINARY PARAMETER SPACE OF DFT-BASED TECHNIQUE

KIICHIRO SAWADA[1] SOUSUKE NAKANO[2]

[1]Prof., Interdisciplinary Faculty of Science and Engineering, Shimane Univ., Matsue, Shimane, JAPAN, kich@riko.shimane-u.ac.jp
[2]Graduate student, Graduate School of Natural Science and Technology, Shimane Univ.,

Editor's Note: This space reserved for the Editor to give such information as date of receipt of manuscript, date of receipt of revisions (if any), and date of acceptance of paper. In addition, a statement about possible written discussion is appended.

DOI: Digital Object Identifier to be provided by Editor when assigned upon publication

ABSTRACT

This paper presents examples of geometric classifications of free-form surfaces in the imaginary parameter space of an initial morphogenesis technique based on Discrete Fourier Transform (DFT), i.e., DFT-based technique. The DFT-based technique can create free-form shells that strictly pass through all the specified control points. Expressive and diverse morphogenesis can be achieved by giving complex values to height coordinates as the control points. It is explained visually and mathematically that the imaginary parts of the complex values play a significant role in determining shape and slope of generated surfaces. It is shown from numerical examples that the geometric shapes of the surfaces are determined by only three parameters of the aforementioned imaginary parts, and geometric classifications of the surfaces are demonstrated as well as mechanical evaluations.

Keywords: *Initial-morphogenesis, Free-form surfaces, Discrete Fourier Transform, Geometric classification*

1. INTRODUCTION

Architectural geometry has significant effects on various aspects, e.g., structural performance, environment and aesthetic properties. Therefore, it is very effective for designers to explore various considerations such as enumeration, comparisons and discussions on the multiple geometric shapes in the initial design stage. In order to conduct the considerations efficiently, it is crucial that various geometric shapes can be represented with as little compound parameter settings as possible. Moreover, it will enable effective considerations if designers can categorize geometry shapes on the parameter space with a few axes. However, there appear to be insufficient efficient methodologies for such considerations. Although NURBS [1-4] succeeds in generating parametric surfaces or curves, it requires slightly more various parameter settings such as knot vectors, weights, and control points. On the other hand, there exists an initial morphogenesis technique by Discrete Fourier Transform (DFT) [5], i.e., DFT-based technique, using very simple parameter settings. This paper presents that expressive morphogenesis can be conducted by giving complex values to height coordinates as the control points on the DFT-based technique and explains visually and mathematically that the imaginary parts of the complex values play a significant role in defining shape and slope of free-form surfaces. It is shown from numerical examples that the geometric shapes of the surfaces are determined by only three parameters of the imaginary parts, and geometric classifications of the surfaces are demonstrated as well as mechanical evaluations.

2. DFT-BASED TECHNIQUE

DFT [7] is applied for various engineering problems [6] such as seismic response [8], audio applications [9], topology optimization [11-13] and morphogenesis [5,10]. The DFT has a lot of strong points such as simple differential formulation, require fewer parameter settings and periodical shape generations [6]. Figure 1 shows a procedure of

the DFT-based technique [5]. After specifying the values of the control point coordinates, DFT is performed on the sequence or matrix of the control point information, and zeros are added as higher-order components to the generated DFT sequence or matrix [5]. Finally, by inverse DFT, morphology can be synthesized with relatively low-order sine waves [5].

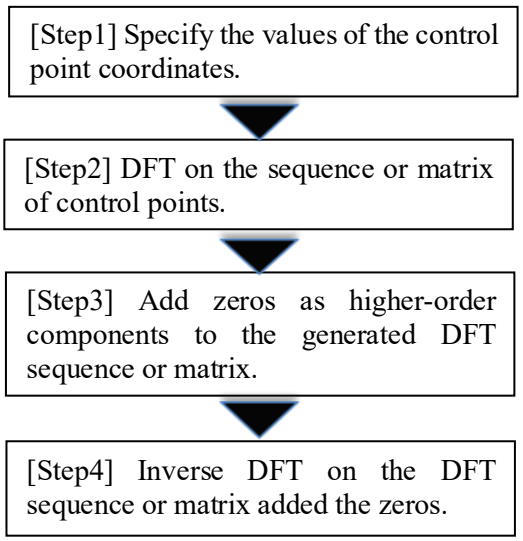


Figure 1: *A Procedure of the DFT-based technique [5]*

3. COMPARISON BETWEEN REAL AND COMPLEX VALUES AS CONTROL POINTS COORDINATES ON SURFACES

Figure 2 shows a generated surface (20 × 20 grid red dots) with real values as control point coordinates (blue circles at four corners) on the DFT based technique. The number of the control points is set to $n_{fx} = 2$ in the x direction and $n_{fy} = 2$ in the y direction, and height coordinates of the control points are set to

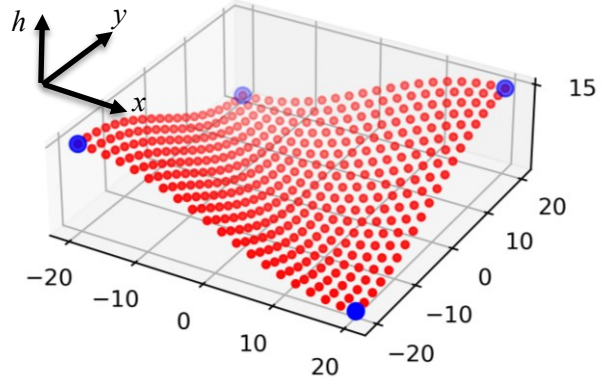


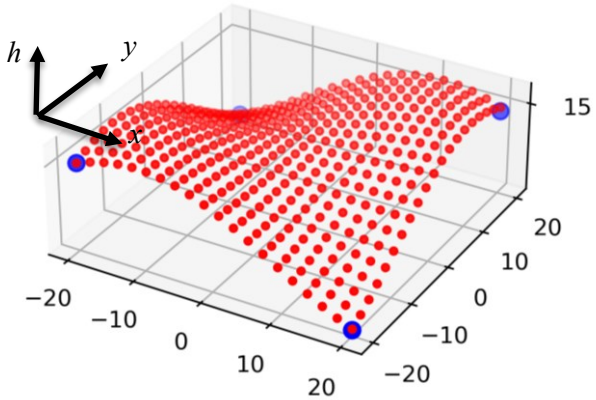
Figure 2: *Generated surface by DFT-based technique with real values as control points coordinate*

$\mathbf{Z} = [Z_{0,0}, Z_{0,1}, Z_{1,1}, Z_{1,0}] = [15.0, 3.0, 15.0, 3.0]$ with a counterclockwise order from the third quadrant point on the x-y plane. Figures 3(A) and (B) show curved surfaces with complex values as height coordinates, $\mathbf{Z} = [15.0+0.0j, 3.0+10.0j, 15.0+20.0j, 3.0+10.0j]$ and $\mathbf{Z} = [15.0+0.0j, 3.0+10.0j, 15.0+10.0j, 3.0+20.0j]$, respectively. These figures indicate the following aspects.

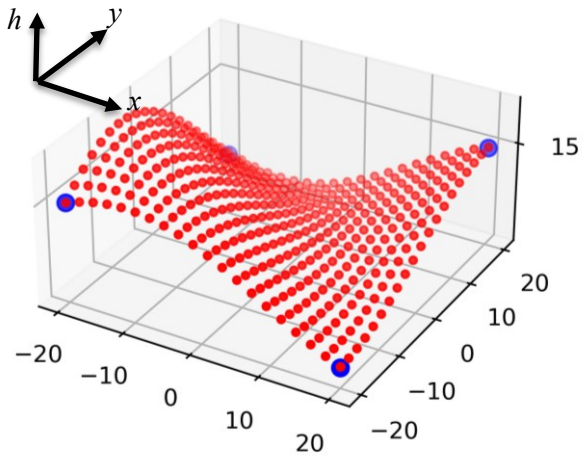
(i)All three surfaces pass through the common four control points.

(ii)The imaginary parts of **Z** give significant effects on the shape of the generated surface even though the real parts are the same.

(iii)It seems that the shape effects are primarily in the slopes of the generated surfaces, and a variety of expressive shapes can be obtained with very simple parameter settings.



(A) $\mathbf{Z} = [15.0+0.0j, 3.0+10.0j, 15.0+20.0j, 3.0+10.0j]$



(B) $\mathbf{Z} = [15.0+0.0j, 3.0+10.0j, 15.0+10.0j, 3.0+20.0j]$

Figure 3: *Generated surface by DFT-based technique with complex values as control points coordinate*

4. MATHEMATICAL FORMULATIONS

The previous section illustrates the role of imaginary parts of **Z** on DFT-based surfaces by the figures. Recent work [6] has already discussed the role of imaginary parts on DFT-based curves. This section discusses the role of the imaginary parts by mathematical formulations for DFT-based surfaces. At first, we can decompose the complex values as the control points coordinates, Z_{KM}, into the real part, $A_{K,M}$, and the imaginary part, $C_{K,M}$, as follows.

$$Z_{K,M} = A_{K,M} + j\, C_{K,M} \quad (K=0,\ldots,n_{fx}\text{-}1,\ M=0,\ldots,n_{fy}\text{-}1) \quad (1)$$

According to Step2 in the above procedure, DFT on **Z** represented by Eq.(1), that is, **z** can be obtained by the linearity of the DFT [6,9] as follows.

$$z_{I,J} = a_{I,J} + j\, c_{I,J} \quad (I=0,\ldots,n_{fx}\text{-}1,\ J=0,\ldots,n_{fy}\text{-}1) \quad (2)$$

, where

$$a_{I,J} = \sum_{K=0}^{n_{fx}-1}\sum_{M=0}^{n_{fy}-1} A_{K,M} e^{-2\pi j(I\cdot\frac{K}{n_{fx}}+J\frac{M}{n_{fy}})} \quad (2a)$$

$$c_{I,J} = \sum_{K=0}^{n_{fx}-1}\sum_{M=0}^{n_{fy}-1} C_{K,M} e^{-2\pi j(I\cdot\frac{K}{n_{fx}}+J\frac{M}{n_{fy}})} \quad (2b)$$

According to Step3 and Step4 in the procedure, the following equation can be obtained by applying the inverse transform on the matrix added zero data strings to $z_{I,J}$.

$$\overline{Z_{X,Y}} = \overline{A_{X,Y}} + j\overline{C_{X,Y}} \quad (X=0,\ldots,n_x\text{-}1,\ Y=0,\ldots,n_y\text{-}1) \quad (3)$$

$$\overline{A_{X,Y}} = \frac{1}{n_{fx}n_{fy}}\sum_{I=0}^{n_{fx}-1}\sum_{J=0}^{n_{fy}-1} a_{I,J} e^{2\pi j(r_{0x}I\cdot\frac{X}{n_x}+r_{0y}J\frac{Y}{n_y})} \quad (3a)$$

$$\overline{C_{X,Y}} = \frac{1}{n_{fx}n_{fy}}\sum_{I=0}^{n_{fx}-1}\sum_{J=0}^{n_{fy}-1} c_{I,J} e^{2\pi j(r_{0x}I\cdot\frac{X}{n_x}+r_{0y}J\frac{Y}{n_y})} \quad (3b)$$

$$r_{0x} = \frac{n_{fx}-1}{n_{fx}}\cdot\frac{n_x}{n_x-1} \quad (3c)$$

$$r_{0y} = \frac{n_{fy}-1}{n_{fy}}\cdot\frac{n_y}{n_y-1} \quad (3d)$$

The above r_{0x} and r_{0y} are the correction coefficients on the inverse DFT [5].

Here, we perform the partial differentiation of Eq.(3) with respect to X and Y, respectively as follows.

$$\frac{\partial\overline{Z_{X,Y}}}{\partial X} = \frac{\partial\overline{A_{X,Y}}}{\partial X} + j\frac{\partial\overline{C_{X,Y}}}{\partial X} \quad (4)$$

$$\frac{\partial\overline{A_{X,Y}}}{\partial X} = \frac{2\pi j r_{0x}/n_x}{n_{fx}n_{fy}}\sum_{I=0}^{n_{fx}-1}\sum_{J=0}^{n_{fy}-1} I\cdot a_{I,J} e^{2\pi j(r_{0x}I\cdot\frac{X}{n_x}+r_{0y}J\frac{Y}{n_y})} \quad (4a)$$

$$\frac{\partial\overline{C_{X,Y}}}{\partial X} = \frac{2\pi j r_{0x}/n_x}{n_{fx}n_{fy}}\sum_{I=0}^{n_{fx}-1}\sum_{J=0}^{n_{fy}-1} I\cdot c_{I,J} e^{2\pi j(r_{0x}I\cdot\frac{X}{n_x}+r_{0y}J\frac{Y}{n_y})} \quad (4b)$$

$$\frac{\partial\overline{Z_{X,Y}}}{\partial Y} = \frac{\partial\overline{A_{X,Y}}}{\partial Y} + j\frac{\partial\overline{C_{X,Y}}}{\partial Y} \quad (5)$$

$$\frac{\partial\overline{A_{X,Y}}}{\partial Y} = \frac{2\pi j r_{0y}/n_y}{n_{fx}n_{fy}}\sum_{I=0}^{n_{fx}-1}\sum_{J=0}^{n_{fy}-1} J\cdot a_{I,J} e^{2\pi j(r_{0x}I\cdot\frac{X}{n_x}+r_{0y}J\frac{Y}{n_y})} \quad (5a)$$

$$\frac{\partial\overline{C_{X,Y}}}{\partial Y} = \frac{2\pi j r_{0y}/n_y}{n_{fx}n_{fy}}\sum_{I=0}^{n_{fx}-1}\sum_{J=0}^{n_{fy}-1} J\cdot c_{I,J} e^{2\pi j(r_{0x}I\cdot\frac{X}{n_x}+r_{0y}J\frac{Y}{n_y})} \quad (5b)$$

Especially under n_{fx} =2 and n_{fy} =2, the real parts of Eq.(4) and Eq.(5) at 4 corners can be obtained as follows.

$$Re\left(\frac{\partial\overline{Z_{X,Y}}}{\partial X}_{(X=0,Y=0)}\right) = \frac{\pi/2}{n_x-1}(C_{1,0}-C_{0,0}) \quad (6a)$$

$$Re\left(\frac{\partial\overline{Z_{X,Y}}}{\partial Y}_{(X=0,Y=0)}\right) = \frac{\pi/2}{n_y-1}(C_{0,1}-C_{0,0}) \quad (6b)$$

$$Re\left(\frac{\partial\overline{Z_{X,Y}}}{\partial X}_{(X=0,Y=n_y-1)}\right) = \frac{\pi/2}{n_x-1}(C_{1,1}-C_{0,1}) \quad (6c)$$

$$Re\left(\frac{\partial\overline{Z_{X,Y}}}{\partial Y}_{(X=0,Y=n_y-1)}\right) = \frac{\pi/2}{n_y-1}(C_{0,0}-C_{0,1}) \quad (6d)$$

$$Re\left(\frac{\partial\overline{Z_{X,Y}}}{\partial X}_{(X=n_x-1,Y=0)}\right) = \frac{\pi/2}{n_x-1}(C_{0,0}-C_{1,0}) \quad (6e)$$

$$Re\left(\frac{\partial\overline{Z_{X,Y}}}{\partial Y}_{(X=n_x-1,Y=0)}\right) = \frac{\pi/2}{n_y-1}(C_{1,1}-C_{1,0}) \quad (6f)$$

$$Re\left(\frac{\partial\overline{Z_{X,Y}}}{\partial X}_{(X=n_x-1,Y=n_y-1)}\right) = \frac{\pi/2}{n_x-1}(C_{0,1}-C_{1,1}) \quad (6g)$$

$$Re\left(\frac{\partial\overline{Z_{X,Y}}}{\partial Y}_{(X=n_x-1,Y=n_y-1)}\right) = \frac{\pi/2}{n_y-1}(C_{1,0}-C_{1,1}) \quad (6h)$$

$C_{1,1}-C_{0,1}$ in Eq.(6c) can be converted to:

$$C_{1,1}-C_{0,1}=(C_{1,1}-C_{0,0})+(C_{0,0}-C_{0,1}) \quad (7)$$

From similar conversions on Eq.(6f), Eq.(6g) and Eq.(6h), it is known that slope variations at 4 corners are determined by just 3 parameters, that is, $C_{0,1}-C_{0,0}$, $C_{1,0}-C_{0,0}$ and $C_{1,1}-C_{0,0}$. In order to fully define a free-form surface at the corners, eight slope components (two per corner) are required. There exist certain limitations for generating surfaces due to only three independent parameters under n_{fx}=2 and n_{fy} = 2. Users should be aware of this point.

5. A GEOMETRIC CLASSIFICATION OF DFT-BASED SURFACES

The previous section explains mathematically that slopes variations at 4 corners are determined by just 3 parameters, that is, $C_{0,1}-C_{0,0}$, $C_{1,0}-C_{0,0}$ and $C_{1,1}-C_{0,0}$ under n_{fx} =2 and n_{fy} =2 on the DFT technique. This section attempts to allocate various surfaces in the space by the three parameters axes. Figure 4(A) represents an example that allocates 27 surfaces generated by the DFT based technique on the three parameters space. The geometric conditions of the surfaces are similar to those of section 3 except for $A_{K,M}$ and $C_{K,M}$. Each surface is obtained by the following settings.

n_{fx} = 2 and n_{fy} =2 ; $C_{0,0}$ is set to a constant (-10).

$C_{0,1}$, $C_{1,0}$ and $C_{1,1}$ are set to either -10, 0 or 10.

$A_{0,0}$, $A_{0,1}$, $A_{1,0}$ and $A_{1,1}$ are set to zero.

A python code for this computation is given in Appendix A1.

Moreover, Fig.4(B) represents another example under the following settings.

n_{fx} = 2 and n_{fy} =2 ; $C_{0,0}$ is set to a constant (0).

$C_{0,1}$, $C_{1,0}$ and $C_{1,1}$ are set to either -10, 0 or 10.

$A_{0,0}$=10, $A_{0,1}$=0, $A_{1,0}$=0 and $A_{1,1}$=10

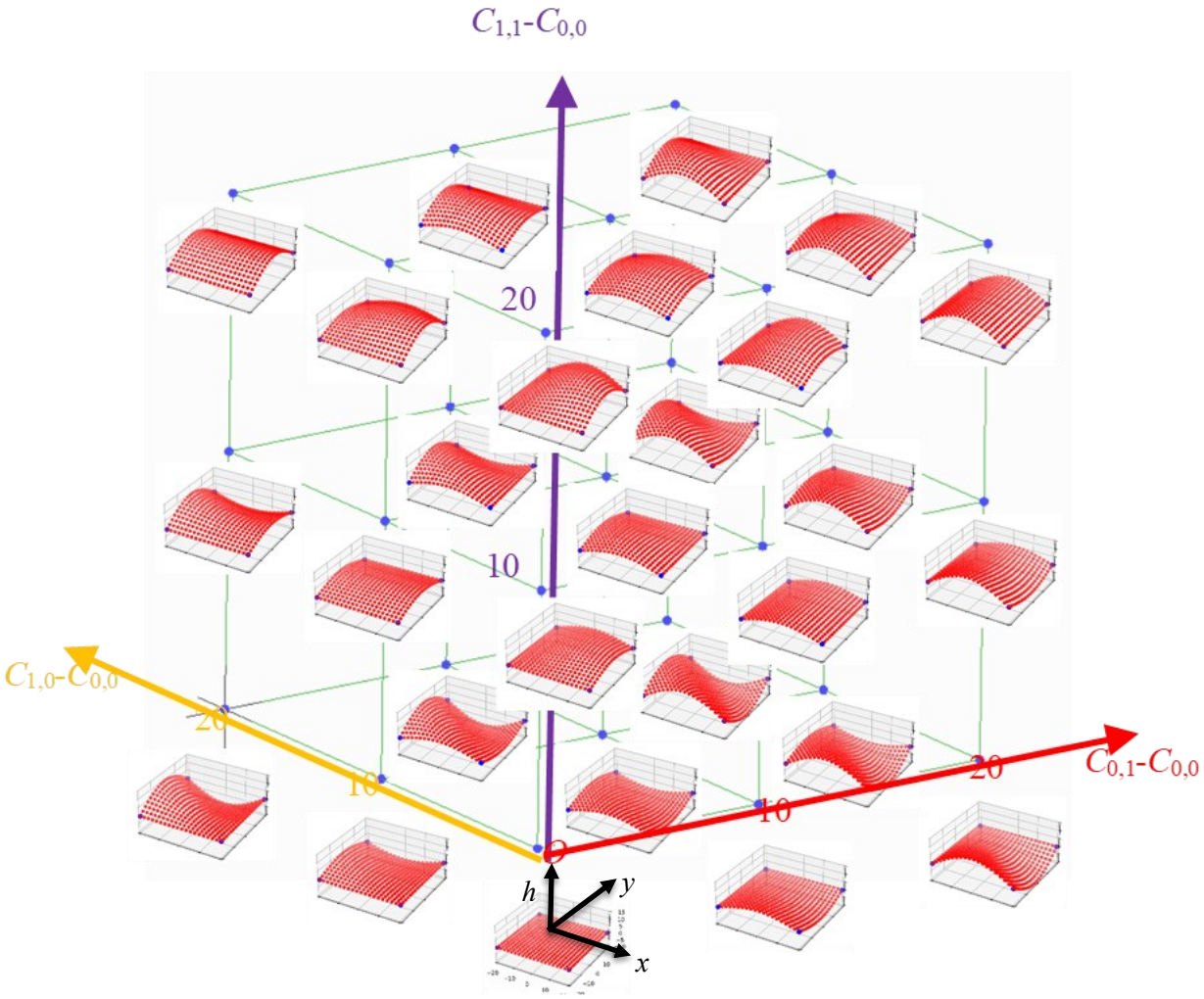


Figure 4(A): *An example of geometric classification of DFT-based surfaces #1*

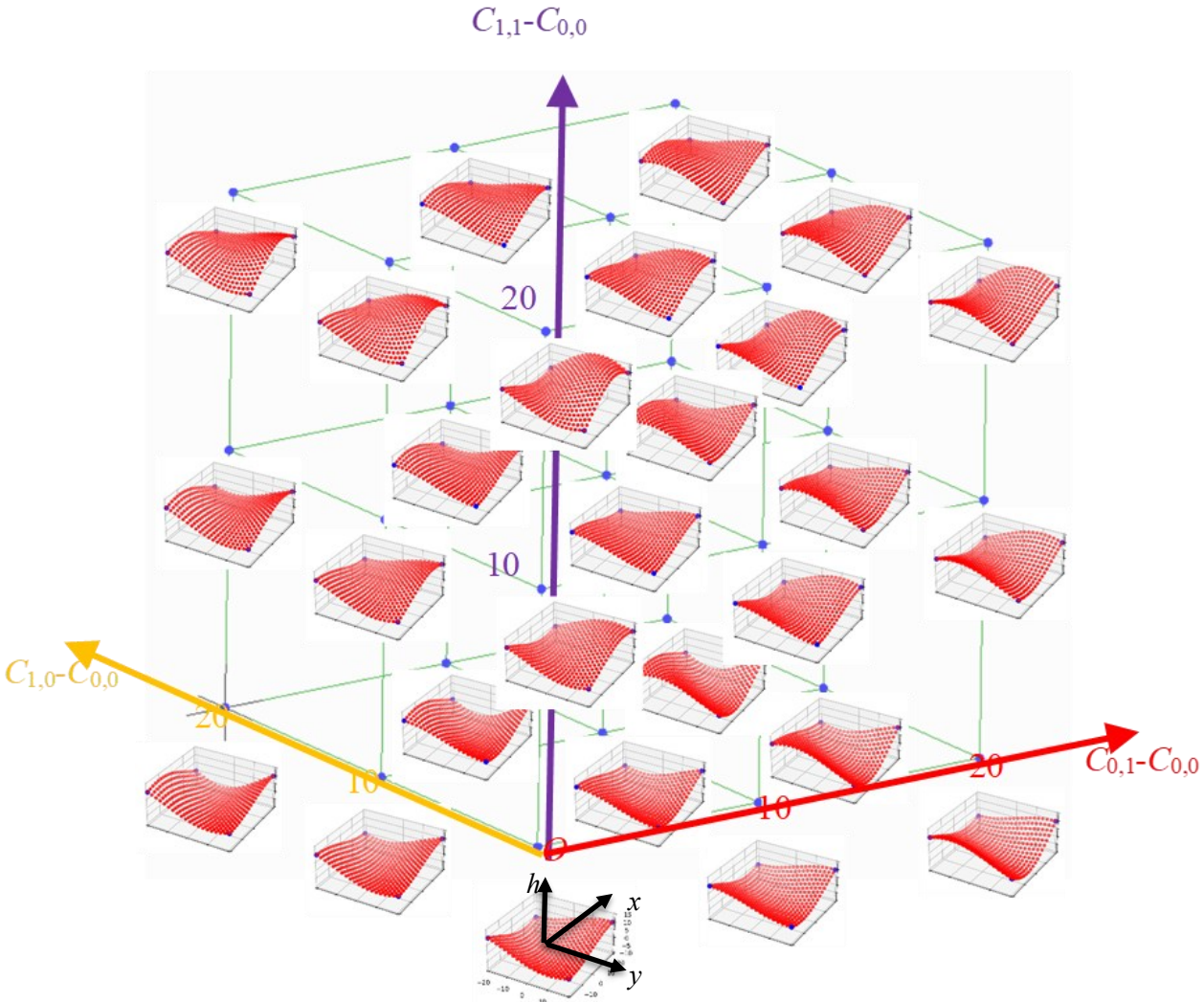


Figure 4(B): *An example of geometric classification of DFT-based surfaces #2*

The following observations are obtained from the figures.

(i) Amount of x direction gradients at the first* and the second* control points increase as $C_{0,1}$-$C_{0,0}$ increases.

(ii) Amount of y direction gradients at the first and the fourth* control points increase as $C_{1,0}$-$C_{0,0}$ increases.

(iii) Amount of x and y direction gradients at the third* control points increase as $C_{1,1}$-$C_{0,0}$ increases.

(iv) It is possible to get an overview of the variations in the morphological properties of the DFT-based surfaces at a glance.

*a counterclockwise order from the third quadrant point on the x-y plane.

The reasons of the remarks (i)-(iii) are obvious from Eqs.(6a)-(6h).

Figure 5(A) represents an example of mechanical evaluations of 41x41 structural models on the cutting plane at $C_{1,1}$-$C_{0,0}$ =20 in fig.4(A). Figure 5(B) presents another example of mechanical evaluations of 41x41 structural models on the cutting plane at $C_{1,1}$-$C_{0,0}$ =10 in fig.4(B). The mechanical evaluations are conducted by computing the vertical displacement that yields the maximum magnitude over all the nodes for each structural model computed under the following common conditions. The displacements are obtained by FEM analysis python library, OpenSeesPy [14,15]. The curved surfaces are modeled as a 20x20 grid of four-node shell elements, that is, ShellMITC4 [16] element object, which uses a bilinear iso-parametric formulation in combination with a modified shear interpolation to improve thin-plate bending performance. Assuming concrete material, Young's modulus of 20000 MPa, Poisson's ratio of 0.2 are used. The four corner edge nodes are fixed in all the

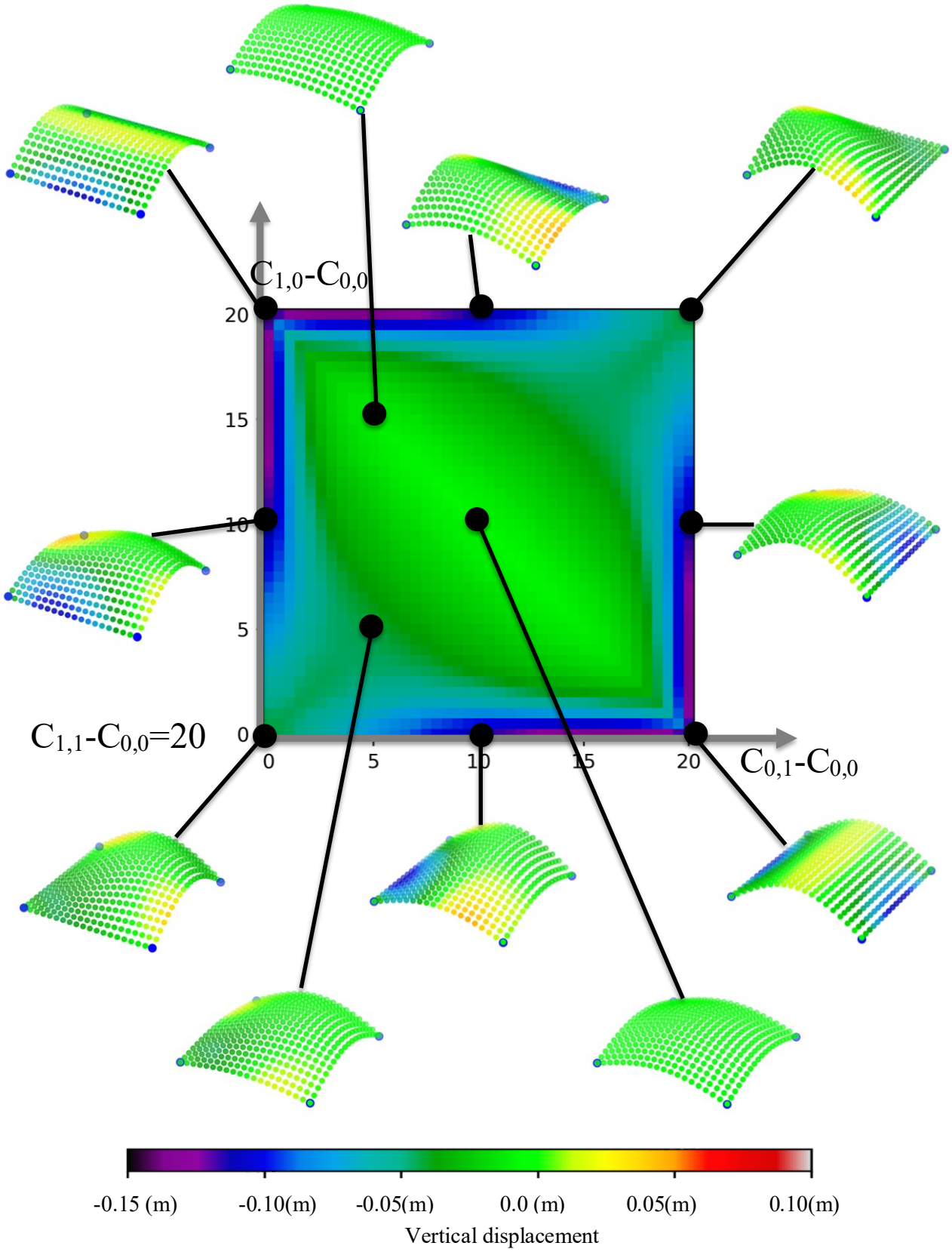


Figure 5(A): *An example of* mechanical evaluations *of DFT-based shells #1 (ShellMITC4)*

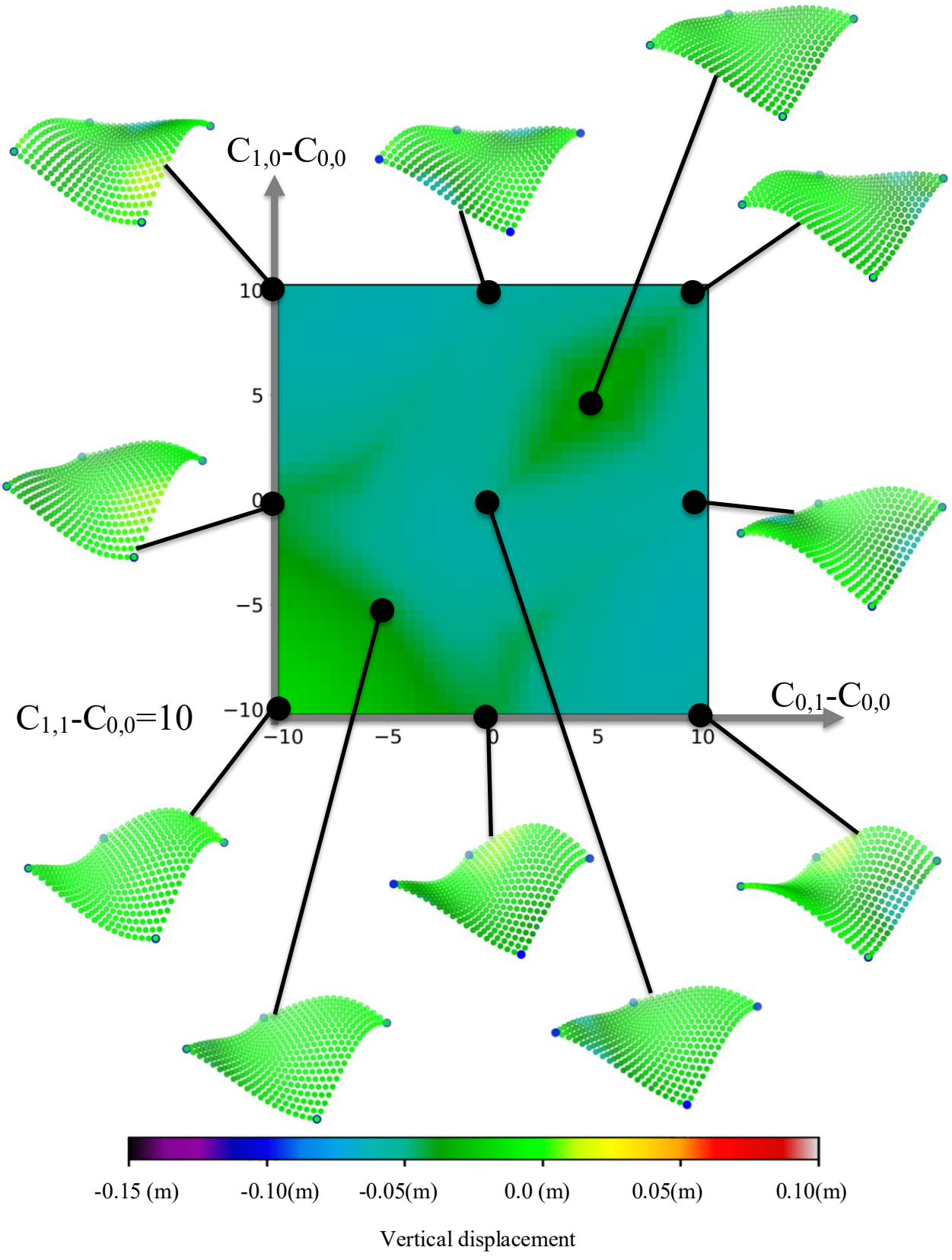


Figure 5(B): *An example of mechanical evaluations of DFT-based shells #2 (ShellMITC4)*

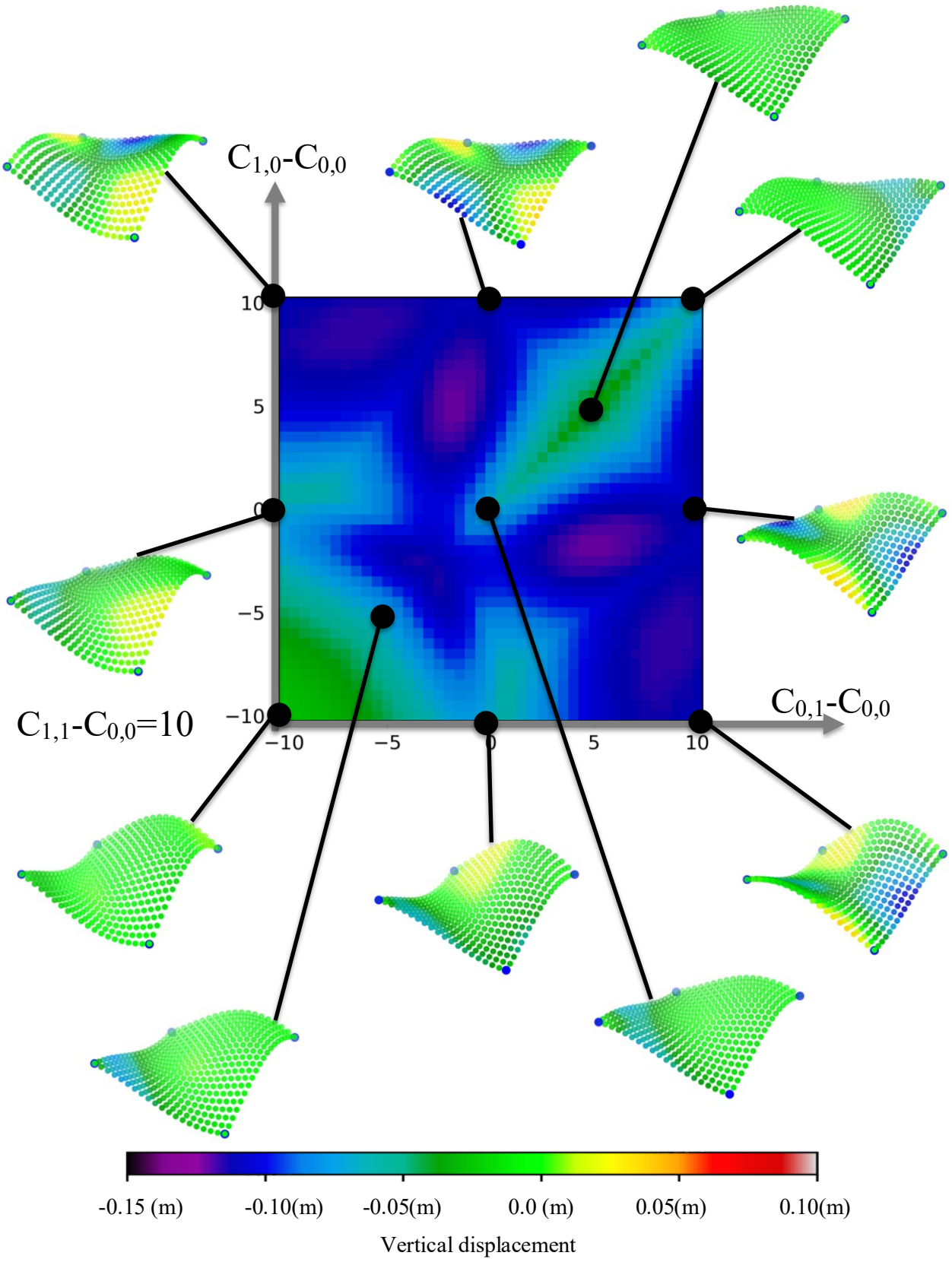


Figure 6: *An example of mechanical evaluations of DFT-based shells #2 (ShellDKGT)*

direction. The shell thickness is 300 mm, and the vertical load of 8000(N) are applied to each node.

Figures 5(A) and (B) also show the vertical displacement distribution over all the nodes for several models.

Tables1(A) and (B) show results using another FEM software or element formulation under same conditions as Figs.5(A) and (B). The first-row result by OpenSeesPy /ShellMITC4 represents the vertical displacement that gives maximum magnitude over all the nodes among all the structural models. The second row onwards by another software or element formulation represents the vertical displacement of same structural model corresponding to the result by OpenSeesPy/ShellMITC4. ShellDKGQ uses two triangular elements generated from the 4 node quadrilateral elements. ShellDKGQ and ShellDKGT in OpenSeesPy are a quadrilateral shell element and a triangular shell element based on the theory of generalized conforming element, respectively [16]. LISA8.0 uses Mindlin thick plate theory which models out-of-plane bending and shear stiffness [17]. Moreover, it uses intermediate nodes generated by linear interpolation between two edged nodes to realize six node triangular or eight node quadrilateral elements.

While there are small differences among five results in Table 1(A), there are distinct differences between triangular element and quadrilateral element in Table 1(B). The reason is that Table 1(B) shows errors due to significant non-coplanarity in the quadrilateral element.

Table 1 (A) Vertical displacement of DFT-based shells #1 by different software

Software	Element shape	Vertical displacement
OpenSeesPy/ ShellMITC4	4 node quadrilateral	-0.134(m)
OpenSeesPy/ ShellDKGQ	4 node quadrilateral	-0.138(m)
OpenSeesPy/ ShellDKGT	3 node triangular	-0.140(m)
LISA8.0	6 node triangular	-0.141(m)
LISA8.0	8 node quadrilateral	-0.150(m)

Table 1 (B) Vertical displacement of DFT-based shells #2 by different software

Software	Element shape	Vertical displacement
OpenSeesPy/ ShellMITC4	4 node quadrilateral	-0.0636(m)
OpenSeesPy/ ShellDKGQ	4 node quadrilateral	-0.0716(m)
OpenSeesPy/ ShellDKGT	3 node triangular	-0.116(m)
LISA8.0	6 node triangular	-0.117(m)
LISA8.0	8 node quadrilateral	-0.0835(m)

Figure 6 represent results with OpenSeesPy/Shell DKGT under same conditions as Fig.5(B). It is observed that the magnitude of the vertical displacement increases over the wide area in Fig. 6 compared to Fig.5(B).

Similar to the consideration of vertical displacement, other evaluations such as principal stress, bending moment information and earthquake resistance are also needed for practical design situations. However, this paper presents just the vertical displacement as an example because the main purpose of the paper is not to discuss mechanical properties of shell structures but to show the possibility of a geometrical classification by just a few imaginary parameters in the DFT based technique.

The following remarks are obtained from the figures.

(i) Variations of $C_{0,1}$-$C_{0,0}$ and $C_{1,0}$-$C_{0,0}$ give significant effects on the shapes of the generated surfaces and the performances under vertical loads.

(ii) Figure 5(A) shows a wide distribution of graph area with relatively good structural performances under vertical loads (green colored area).

(iii) Figure 5(B) and Figure 6 show narrow distribution of graph area with relatively good structural performances under vertical loads (green colored area).

(iv) Since, from Eqs.(6a)-(6h), it is easy to imagine intermediate surface shapes not shown in the figures, it is possible to get an overview of both the structural performances under vertical loads and aesthetics of the surfaces at a glance.

5. CONCLUSIONS

This paper presents an outline of the DFT-based technique for initial morphogenesis of free-form surfaces and explains visually and mathematically that the imaginary parts of the complex values as control points play a significant role in defining shape and slope of DFT-based surfaces. Moreover, the paper provides two examples of geometric classifications in the imaginary parameter space with only three parameters of DFT-based technique as well as mechanical evaluations. The classifications and the mechanical evaluations imply the possibility for architects and engineers to get an overview of the morphological variations and the structural performances of the DFT-based surfaces at a glance. It is expected that the DFT-based technique and classifications of geometric shapes on a few parameters space serve as an effective tool for architects and engineers to explore various considerations, that is, not only vertical displacement shown here but also multiple mechanical and environmental evaluations in initial design stages. This paper presents a geometrical classification under n_{fx} =2 and n_{fy} =2 on the DFT technique. The control points can be set on not only boundary points but also middle points within the scope of the current procedure. Although larger values of n_{fx} and n_{fy} can generate more complex surfaces, the geometric classification might become challenging due to the increased number of parameters. In such cases, optimization techniques will become effective as well as geometric classifications on the cutting plane around the optimum point.

In this way, the presented framework complements traditional parametric approaches such as NURBS by offering a more compact and intuitive parametrization. This can broaden the designer's ability to explore architectural geometry efficiently while maintaining control over form generation in the early design stages.

ACKNOWLEDGMENTS

I would like to express deep gratitude to Assistant Professor NGUYEN TRAN Yen Khang, Shimane Univ. for valuable comments from viewpoints of architectural design and English representation.

REFERENCES

[1] O. R. Bingol and A. Krishnamurthy, "NURBS-Python: An open-source object-oriented NURBS modeling framework in Python," *SoftwareX,* Vol.9, Pages 85-94, January–June, 2019

(https://doi.org/10.1016/j.softx.2018.12.005)

[2] S. Mohan, S. H. Kweon, D. M. Lee, and S. H. Yang, "Parametric NURBS curve interpolators: a review," *International Journal of Precision Engineering and Manufacturing*, 9(2), 84-92, 2008

[3] Y. Okita and T. Honma, "Structural Morphogenesis for Asymmetric Free-Form Grid Shell Using NURBS with Manipulation of Decent Solutions Search," *Proceedings of IASS Annual Symposia,* IASS 2016 Tokyo Symposium: Spatial Structures in the 21st Century – Form Finding & Optimization, pp. 1-9(9)

[4] T. Kimura and H. Ohmori, "Computational Morphogenesis of Free Form Shell," *Journal of the International Association for Shell and Spatial Structures,* Vol. 49, No. 3 December n. 159, pp. 175-180(6), 2008

[5] K. Sawada, "An Initial-Morphogenesis Technique of Free-Form Shell Roofing Based on a Fourier Transform," Journal of the International Association for Shell and Spatial Structures, Vol. 64, No. 3 September n. 217, pp. 199-210(12), 2023

(https://doi.org/10.20898/j.iass.2023.015)

[6] K. Sawada, "An Improved Interpolator Based on a Discrete Fourier Transform For Design Curve Generation," Available at SSRN 4801288.

(http://dx.doi.org/10.2139/ssrn.4801288)

[7] W. C. James and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.* 19: 297-301, 1965

[8] N Makris, M C Constantinou, "Spring-viscous damper systems for combined seismic and vibration isolation," *Earthquake engineering & structural dynamics*, volume 21, issue 8, 649-664, 1992

[9] J. O. Smith, "Mathematics of the discrete Fourier transform (DFT): with audio applications," *CCRMA*, Stanford, July 2002

[10] Y. Toyoshima, S. Yamada, K. Sato, T. Nagai and M. Araya, "Study on Structural Morphogenesis by Applying of Fourier Series to Circular Structure," *Summaries of Technical Papers of Annual Meeting,* AIJ, pp.835-836, 2009. (in Japanese)

[11] K. Sawada, "Topology optimization of large deformable elastic plates," *Journal of Structural and Construction Engineering (Transactions of AIJ),* Vol.85, No.771, pp.683-692, 2020.5 (in Japanese)

(https://doi.org/10.3130/aijs.85.683)

[12] *K. Sawada,* "Seismic Response Analyses of RC Portal Frames with Large Deformable Elastic Braces." *Seismic Resistant Structures*: 197, 2018

(https://doi.org/10.2495/CMEM-V6-N5-880-886)

[13] K. Sawada, K. Kajitani, T. Uno, J. Teramoto, S. Komatsu, "A Study on Multi-Objective Optimization of Large Deformable Elastic Plates," *Buildings*, 12(9), 1323, 2022

(https://doi.org/10.3390/buildings12091323)

[14] Zhu, M., McKenna, F., & Scott, M. H., "OpenSeesPy: Python library for the OpenSees finite element framework," *SoftwareX*, 7, 6-11, 2018

(https://doi.org/10.1016/j.softx.2017.10.009)

[15] McKenna, F., Scott, M. H., & Fenves, G. L., "Nonlinear finite-element analysis software architecture using object composition," *Journal of computing in civil engineering*, 24(1), 95-107, 2010

(https://doi.org/10.1061/(ASCE)CP.1943-5487.0000002)

[16] https://openseespydoc.readthedocs.io/en/latest/index.html (Final Access Sep 27, 2025)

[17] Tutorials and Reference Guide in LISA8.0

https://www.lisafea.com/

(Final Access Sep 27, 2025)

APPDENDIX A1: PYTHON CODE-SECTION 5

A Python code is created by the authors as follows.

After you copy this code to any frameworks, please delete any unnecessary spaces following the two backslashes in the code before the computation.

```
#DFT_surface_demo Mon Sep 27 18:40 2025 by Kiichiro SAWADA and Sosuke NAKANO
import numpy as np
import matplotlib.pyplot as plt


nx = 20 ; ny = 20 # number of nodes in x and y direction
nfx = 2 ; nfy = 2 # number of control points in x and y direction
lx = 20.0 ; ly = 20.0 # half span of shell surfaces in x and y direction
# r_r:real part of control points height cooridnates
r_r = np.array([0.0,0.0,0.0,0.0])
# r_i:imaginary part of control points height coordinates
r_i = np.array([0.0j,0.0j,0.0j,0.0j])
c0 = -10.0j ; c1 = np.array([-10.0,0.0,10.0])
c2 = np.array([-10.0,0.0,10.0]) ; c3 = np.array([-10.0,0.0,10.0])


for ii in range (0,3):
    for kk in range (0,3):
        for ll in range(0,3):
            r_i = [c0, c1[ii]*1.0j, c2[kk]*1.0j, c3[ll]*1.0j]
            a = np.linspace(0,nx-1,nx) ; b = np.linspace(0,ny-1,ny)
            ia = np.int16(a[:]) ; ib = np.int16(b[:])
            iaa,ibb = np.meshgrid(ia,ib)
            x = np.linspace(-lx, lx, nx) ; y = np.linspace(-ly, ly, ny)
            x_c = np.linspace(-lx, lx, nfx) ; y_c = np.linspace(-ly, ly, nfy)
            X, Y = np.meshgrid(x, y) ; X_c, Y_c = np.meshgrid(x_c, y_c)


            def ifftcon(zfa):
                r0x = (nfx-1)/nfx*nx/(nx-1)
                r0y = (nfy-1)/nfy*ny/(ny-1)
                izfs = np.zeros([ny,nx],dtype='complex128')
                for j in range(nx):
                    for i in range(ny):
                        izfs = izfs + zfa[i,j]*np.exp(2.0j*np.pi*\
                        (j*iaa[:,:]/nx*r0x+i*ibb[:,:]/ny*r0y))
                izfs = izfs/nfx/nfy ; izfsr = izfs.real
                return izfsr
# [Step1] Specify the value of the control point coordinates.
            r = r_r + r_i
            Z_c = r.reshape(nfy,nfx).real ; zf0 = r.reshape(nfy,nfx)
# [Step2] DFT on the sequence or matrix of control points.
            zf1 = np.fft.fftn(zf0)
# [Step3] Add zeros as higher-order components.
            zfa = np.array([[zf1[i,j] if i<nfy and j<nfx else 0.0 \
            for j in range(nx)] for i in range(ny)])
# [Step4] Inverse DFT on the DFT sequence or matrix added the zeros.
            Z = ifftcon(zfa)
# Plot figures
            fig = plt.figure(dpi=400) ; ax = fig.add_subplot(projection='3d')
            ax.set_box_aspect((40,40,20))
            ax.set_xlim(-20,20) ; ax.set_ylim(-20,20) ; ax.set_zlim(-10,15)
            #ax.tick_params(labelbottom=False, labelleft=False)
            ax.scatter(X,Y,Z, color='red')
            ax.scatter(X_c,Y_c,Z_c,s=50, color='blue') ; plt.show()
```