

DOCTORAL THESIS

**HIGH PERFORMANCE OF THE
GENERALIZED FINITE DIFFERENCE
METHOD AND APPLICATIONS**

Augusto César Albuquerque Ferreira

Salamanca, 2022



**VNIVERSIDAD
D SALAMANCA**

CAMPUS DE EXCELENCIA INTERNACIONAL

HIGH PERFORMANCE OF THE GENERALIZED FINITE DIFFERENCE METHOD AND APPLICATIONS

A thesis submitted in fulfilment of the requirements for the degree of Doctor por la Universidad de Salamanca in the Department of Computer Engineering under the research line of numerical methods

Augusto César Albuquerque Ferreira

SUPERVISORS:

Miguel Ureña Asensio

Higinio Ramos Calle

Declaration

I hereby declare that the thesis entitled "*HIGH PERFORMANCE OF THE GENERALIZED FINITE DIFFERENCE METHOD AND APPLICATIONS*" being submitted to the University of Salamanca, Spain, for the award of the degree of **Doctor por la Universidad de Salamanca** contains the original work carried out by me under the supervision of Prof. Miguel Ureña Asensio and Prof. Higinio Ramos Calle.

The research work reported in this thesis is original and has not been submitted either in part or full to any university or institution for the award of any degree or diploma.

Salamanca, 17 June 2022

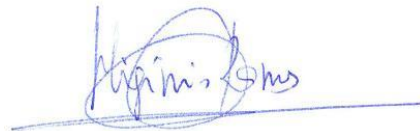


Augusto Albuquerque Ferreira
(Research scholar)

V.B. (Approval)



Miguel Ureña Asensio
(Supervisor)



Higinio Ramos Calle
(Supervisor)

HIGH PERFORMANCE OF THE GENERALIZED FINITE DIFFERENCE METHOD AND APPLICATIONS

Summary

We solve 2D and 3D second-order partial differential equations considering the Generalized Finite Difference Method (GFDM) with third- and fourth-order approximations.

First of all, we analyze the influence of the number of points per star and establish some values as references.

Secondly, we propose a new strategy to deal with ill-conditioned stars, which are frequent in higher-order approximations. This strategy uses a few points per star in relation to those established as reference and presents excellent results for detecting ill-conditioned stars, increasing the accuracy of the numerical approximation and reducing the computational cost.

To implement the algorithm, we use good programming practices together with higher-order approximations in the GFDM to reduce the computational cost at different stages of the calculation.

On the other hand, we have developed a strategy to obtain discretizations adapted to the specific problem to be solved. This strategy distributes the points in the domain according to the gradient values, which allows using a discretization with a smaller number of points, reducing the computational cost and maintaining the accuracy that would be achieved with finer discretizations where the points are distributed homogeneously.

Furthermore, we develop a 3D adaptive algorithm with fourth-order approximations on irregular initial discretizations. We compare the results with the algorithm of points added halfway. In all applications, we achieve better accuracy with a decrease in the final number of points and computational time.

Finally, to test the performance of the algorithm in a real problem, we evaluate the seismic responses in onshore wind turbines using the GFDM coupled with the Newmark method. We compare the history of transversal displacement with a model based on the Finite Element Method using the ABAQUS software. The results are essentially identical and show the validity of the model proposed in the GFDM.

ALTO RENDIMIENTO DEL MÉTODO DE LAS DIFERENCIAS FINITAS GENERALIZADAS Y APLICACIONES

Sumario

En esta tesis se aborda la resolución de ecuaciones en derivadas parciales de segundo orden en 2D y 3D por el Método de las Diferencias Finitas Generalizadas (MDFG) utilizando aproximaciones de tercer y cuarto orden.

En primer lugar, se analiza la influencia del número de puntos por estrella y se establecen algunos valores a modo de referencia.

En segundo lugar, se ha desarrollado una nueva estrategia para detectar y tratar estrellas mal condicionadas, las cuales pueden aparecer con frecuencia cuando se utilizan aproximaciones de orden superior. Esta estrategia utiliza una cantidad de puntos por estrella menor que los establecidos como referencia y presenta excelentes resultados detectando estrellas mal condicionadas, aumentando la precisión de la aproximación numérica y reduciendo el coste computacional.

Para implementar el algoritmo, se han utilizado buenas prácticas de programación junto con las aproximaciones de orden superior en el MDFG para reducir el coste computacional en diferentes etapas del cálculo.

Por otro lado, se ha desarrollado una estrategia para obtener discretizaciones adaptadas al problema concreto que se desea resolver. Esta estrategia distribuye los puntos en el dominio conforme a los valores del gradiente, lo que permite usar una discretización con un menor número de puntos, reduciendo así el coste computacional y manteniendo la precisión que se alcanzaría con discretizaciones más finas donde los puntos se distribuyen más homogéneamente.

Además, se ha desarrollado un algoritmo adaptativo para problemas en 3D con aproximaciones de cuarto orden a partir de discretizaciones iniciales irregulares. Se han comparado los resultados del algoritmo propuesto con los del algoritmo de puntos

añadidos a media distancia. En todas las aplicaciones, se ha conseguido una mayor precisión junto con una disminución del número final de puntos y del tiempo computacional.

Finalmente, para probar el desempeño del algoritmo en un problema real se ha evaluado la respuesta sísmica en aerogeneradores terrestres empleando el MDFG acoplado con el método de Newmark. Se han comparado los datos del desplazamiento transversal con un modelo basado en el método de los elementos finitos utilizando el programa ABAQUS. Los resultados son esencialmente idénticos y muestran la validez del modelo propuesto en el MDFG.

Acknowledgements

I thank God first and foremost for the completion of this Thesis.

This research was supported by the international scholarship program of the University of Salamanca-Santander Bank.

I would like to express my sincere gratitude to my supervisors Dr. Miguel Ureña and Dr. Higinio Ramos. They were fundamental for the realization of this thesis.

I acknowledge with a deep sense of reverence, my gratitude towards my parents for their enormous support and encouragement.

Finally, I thank my bride Thaís Rodrigues de Albuquerque for her constant company at all times.

Contents

1	Introduction	31
1.1	State of the art	31
1.1.1	The generalized finite difference method	31
1.1.2	Higher-order approximations	33
1.1.3	Computational aspects	35
1.1.4	Treatment of ill-conditioned stars	35
1.1.5	Different forms of discretization using the GFDM	36
1.1.6	Adaptive methods using the GFDM	37
1.1.7	Simplified models in the seismic response of wind turbines and the GFDM in dynamic responses	39
1.2	Objectives	40
2	The generalized finite difference method	43
2.1	Solution of a Dirichlet boundary value problem by the GFDM	48
2.2	Influence of the number of points per star with higher-order approxi- mations	49
2.2.1	Approximations in 2D	50
2.2.2	Approximations in 3D	53
3	Computational aspects	57
3.1	Parallel process in the GDFM	59
3.2	A vectorized algorithm in the GFDM	59
3.3	Examples	63

3.3.1	Speedup with parallel process	63
3.3.2	Times with vectorized algorithm	64
4	Treatment of ill-conditioned stars	67
4.1	Strategy to avoid ill-conditioned stars	67
4.2	Numerical results	69
4.2.1	Results with the proposed strategy to avoid ill-conditioned stars in 2D	70
4.2.2	Results with the proposed strategy to avoid ill-conditioned stars in 3D	73
4.2.3	Influence of the weighting function	75
4.2.4	Influence of the number of intervals on the proposed strategy .	76
5	A technique for generating adapted discretizations	79
5.1	A procedure for generating discretizations adapted to the partial dif- ferential equation	79
5.1.1	Procedure for generating interior points	80
5.1.2	Procedure for generating boundary points	83
5.2	Numerical results	84
5.2.1	Adapted discretizations with fourth-order approximation . . .	85
5.2.2	Influence of the parameters η and β	94
5.2.3	Influence of the weighting functions	95
5.2.4	Adapted discretizations with second-order approximations . .	96
6	An h-adaptive method in 3D	99
6.1	Error indicator	99
6.2	An adaptive procedure	99
6.3	Procedure for adding boundary points	101
6.4	Examples	102
6.4.1	Example 1	102
6.4.2	Example 2	104

6.4.3	Example 3	105
6.4.4	Example 4	106
6.4.5	Example 5	107
7	Seismic response in onshore wind turbines	109
7.1	Partial differential equation of motion - beam bending	109
7.2	Analysis of dynamic response	111
7.3	Procedure to calculate the natural vibration modes	115
7.4	Turbine NREL	117
7.4.1	External force applied - earthquake	117
7.4.2	Result of dynamic response	118
7.5	Turbine Senvion MM92	120
7.5.1	External force applied - earthquake	121
7.5.2	Result of dynamic response	122
8	Conclusions and future lines of research	125
8.1	Conclusions	125
8.2	Future developments	130
9	Conclusiones y futuras líneas de investigaciones	133
9.1	Conclusiones	133
9.2	Desarrollos futuros	139

List of Figures

2-1	Discretization of the domain and the star formation in 3D (h_i, k_i and l_i are the relative distances of \mathbf{x}_i to \mathbf{x}_0)	45
2-2	The domains and discretizations used in the 2D examples.	50
2-3	Numerical errors for each approximation order in 2D	52
2-4	The domains and discretizations used in the 3D examples.	53
2-5	Numerical errors for each order of approximation in 3D applications	55
4-1	Strategy to avoid large condition number in the matrix \mathbf{A}	68
4-2	Distribution of the formation of stars used at each point in the discretization: stage 1 (quadrant criterion) and stage 2 (quadrant criterion with the addition of points).	72
5-1	Influence areas for each point in the domain with their geometric centres at the respective points. Note that there is no overlap between the influence areas relative to the internal points. However, overlap can occur between the influence area of a interior point and the influence area of a boundary point.	81
5-2	The insertion of points within the areas of influence (left) and the discretization smoothing process (right).	82

5-3	Interior points generated from boundary points and boundary points inserted. The points p1, p2, p3, and p4 have the closest interior areas A1, A2, A3, and A4 with 4, 1, 1, and 9 points, respectively. The white points are rejected because they are outside the domain, the yellow point is removed because it is too close to the boundary, and the green point is removed because it is close to an interior point generated from a boundary point. The blue and red points are the interior and boundary points inserted, respectively.	83
5-4	Gradients, adapted and uniform discretizations for Example 1.	86
5-5	Gradients, adapted and uniform discretizations for Example 2.	87
5-6	Gradients, adapted and uniform discretizations for Example 3.	89
5-7	Gradients, adapted and uniform discretizations for Example 4.	90
5-8	Gradients, adapted and uniform discretizations for Example 5.	92
5-9	Gradients, adapted and uniform discretizations for Example 6.	93
5-10	Relationship between the value of η and the global errors.	94
5-11	Relationship between the value of β and the global errors.	95
5-12	Relationship between the value of η , weighing functions, and global errors.	96
5-13	Adapted and uniform discretizations for Examples 1 and 2 with second-order approximations	97
6-1	Position of the added points in relation to the refined point. On the left, 8 points are added; and on the right, 14 points are added.	101
6-2	Discretizations in the adaptive algorithm for Example 1.	104
6-3	Discretizations in the adaptive algorithm for Example 2.	105
6-4	Discretizations in the adaptive algorithm for Example 3.	106
6-5	Discretizations in the adaptive algorithm for Example 4.	107
6-6	Discretizations in the adaptive algorithm for Example 5.	108

7-1	Euler-Bernoulli beam with a top mass and cross section variable subjected to a horizontal earthquake. The soil-structure interaction is modelled by two flexible springs.	110
7-2	Discretization with the GFDM and sparsity pattern of matrices \mathbf{K} and \mathbf{M}	113
7-3	Accelelogram for the Loma Prieta earthquake.	118
7-4	History of transversal displacement considering soil-structure interaction and damping ratio in the turbine NREL-5 MW.	120
7-5	Accelelogram for the Cape Mendocino earthquake.	122
7-6	History of transversal displacement considering soil-structure interaction and damping ratio in the turbine Senvion MM92.	123

List of Tables

3.1	Summary of the algorithm developed in GFDM.	63
3.2	Speedup of each calculation process in the 2D case.	64
3.3	Speedup of each calculation process in the 3D case.	64
3.4	The runtime of the 2D code to calculate the derivatives and assemble the \mathbb{K} matrix.	65
3.5	The runtime of the 3D code to calculate the derivatives and assemble the \mathbb{K} matrix.	65
3.6	Times to the non-vectorized and vectorized codes in 2D and 3D examples.	66
4.1	Global errors with the third-order approximation in 2D	71
4.2	Global errors with the fourth-order approximation in 2D	71
4.3	Global error with third-order approximation in 2D for the discretiza- tion ID1.	73
4.4	Global error with fourth-order approximation in 2D for the discretiza- tion ID1.	73
4.5	Global error with the third-order approximation in 3D	74
4.6	Global error with the fourth-order approximation in 3D	74
4.7	Global error with the third-order approximation in 3D for the dis- cretization ID4.	75
4.8	Global error with the fourth-order approximation in 3D for the dis- cretization ID4.	75
4.9	Relation between the weighting functions, the global error and the percentage (%) of modified stars.	76

4.10	Relation between the number of intervals λ , the global error and the percentage (%) of modified stars.	77
5.1	Global error and execution times for Example 1	86
5.2	Global error and execution times for Example 2	87
5.3	Global error and execution times for Example 3	89
5.4	Global error and execution times for Example 4	91
5.5	Global error and execution times for Example 5	92
5.6	Global error and execution times for Example 6 with fourth-order approximations.	93
5.7	Data for different weighting functions ($\eta = 0.25, \beta = 0.5$).	96
5.8	Global error and execution times for Example 1 with second-order approximation	98
5.9	Global error and execution times for Example 2 with second-order approximation	98
6.1	Coordinates of the points to be added for each point (x_i, y_i, z_i) to be refined (e_i is the error indicator; \bar{e} and σ_e are the average and the standard deviation of the error indicators, respectively; Δ is the shortest distance between two points in all the domain).	100
6.2	Errors, number of points, number of adaptive steps, and time in each algorithm for Example 1.	103
6.3	Errors, number of points, number of adaptive steps, and the time in each algorithm for Example 2.	105
6.4	Errors, number of points, number of adaptive steps, and the time in each algorithm for Example 3.	106
6.5	Errors, number of points, number of adaptive steps, and the time in each algorithm for Example 4.	107
6.6	Errors, number of points, number of adaptive steps, and the time in each algorithm for Example 5.	108

7.1	Summary of Baseline Wind Turbine Properties (NREL- 5 MW) . . .	118
7.2	Modal frequencies to the Bmodes, FEM and GFDM in the turbine NREL	119
7.3	Summary of Baseline Wind Turbine Properties (Senvion MM92- 2,050 kW)	121
7.4	Modal frequencies in the FEM and the GFDM to the turbine Senvion MM92.	122

List of Symbols and acronyms

D	Domain
Φ	Dimension
Ω	Interior domain
\mathbf{x}	Spatial coordinates
M	Discretization of the domain
N	Total number of points
m	Number of star points
V	Set of star points
\mathbf{x}_0	Coordinate at the central point
\mathbf{x}_i	Coordinate at a point of the star
U	Any function determined on the domain D
r	Residual vector
u_0	Approximate value at the central point
u_i	Approximate value at a point of the star
\mathbf{u}	Approximate value vector
$\Delta\mathbf{u}$	Difference vector
\mathbf{P}	Matrix of coefficients of the Taylor series
\mathbf{D}_u	Vector of approximate derivatives at the central point
h	Relative distance in relation to the x axis
k	Relative distance in relation to the y axis
l	Relative distance in relation to the z axis

\mathbf{W}	Diagonal matrix of weights
\mathbf{A}	Matrix of coefficients the GFDM
\mathbf{B} and $\bar{\mathbf{B}}$	Matrices of coefficients of the derivatives
$b_{i,j}$	Elements of the \mathbf{B} matrix
p	Number of terms of the Taylor series
\mathcal{L}	Second-order linear differential operator
f and g	Known real functions
a_0	Constant
\mathbf{a}	Coefficients of the partial derivatives
\mathbb{K}	Coefficient matrix of the global equation system
NI	Total number of interior points
\mathbf{F}	Vector of the values of f at each interior point
$\bar{\mathbf{F}}$	Vector \mathbf{F} with boundary conditions applied
$\tilde{\mathbf{u}}$	Vector of approximate values of U at the points of M
$\bar{\mathbf{u}}$	Vector of approximate values of U at the interior points of M
$\mathbf{B}_g, \mathbf{I}_g, \mathbf{J}_g$ and \mathbf{a}_g	four global 1D-arrays
\mathbf{d}	Vector of minimum distances
λ	Number of intervals
δ_i	Length of an interval
d_{min}	Minimum value of \mathbf{d}
d_{max}	Maximum value of \mathbf{d}
$\kappa(\mathbf{A})$	Condition number of \mathbf{A}
tol	Tolerance
Error	Global error
m_0	Initial number of points per star
$\vec{\nabla}_0$	Vector gradient at each interior central point
Δ	Shortest distance between two points in all the domain

$\bar{\nabla}$	Average gradient
σ	Standard deviation of the gradients
β	Constant
η	Expansion percentage
α	Percentage of broadening of the regular grid in the influence area
n	Number of points within the area of influence in one direction
e_i	Error indicator at a point
\bar{e}	Mean of the error indicators
σ_e	Standard deviation of the error indicators
$\tilde{\mathbf{P}}$	3D array with all \mathbf{P} matrices of the stars
$\tilde{\mathbf{W}}$	3D array with all \mathbf{W} matrices of the stars
$\mathbf{h}_i, \mathbf{k}_i,$ and \mathbf{l}_i	1D arrays with all relative distances
$\tilde{\mathbf{B}}$	3D array with all \mathbf{B} matrices of the stars
\mathbb{K}_s	Global matrix formed by the coefficients of the stars
\mathbb{K}_o	Global matrix formed by the coefficients of central points
$\mathbf{Bo}_g, \mathbf{Io}_g,$ and \mathbf{Jo}_g	1D-arrays that generate \mathbb{K}_o
$\mathbf{Bs}_g, \mathbf{Is}_g,$ and \mathbf{Js}_g	1D-arrays that generate \mathbb{K}_s
\mathbf{ao}_g and \mathbf{as}_g	1D-arrays formed from the coefficients of the differential equation
$S_1^k \dots S_m^k$	Global numbering of the stars
ν	Exact transverse displacement
E	Young's modulus
I	Cross-sectional moment of inertia
t	Time
f_g	Transversal force along the structure
μ	Structural mass per unit length
Q	Axial force along the structure
M_t	Concentrated mass
J	Rotary inertia

g	Gravitational acceleration
k_t	Foundation lateral stiffness
k_r	Foundation rocking stiffness
a_g	Horizontal acceleration caused by an earthquake at the base of the structure
f_t	Concentrated force at the top of the structure
\hat{a}_t	Approximate acceleration in at top of the tower
\hat{a}_0	Approximate acceleration in at central point
L	Height of the tower
\hat{v}	Approximate value of transverse displacement
\hat{a}	Approximate value of transverse acceleration
f_0	Force at the central point
\hat{v}	Approximate value of transverse displacement at the central point
\hat{v}_i	Approximate value of transverse displacement at the star points
μ_0	Structural mass per unit length at the central point
I	Cross-sectional moment of inertia at the central point
\hat{v}	Transverse displacement vector
\hat{a}	Transverse acceleration vector
Fg	Force vector
C	Damping matrix
ω	Frequency matrix
c_1 and c_2	Proportionality constants
ξ	Damping ratio
ω	Natural frequency
Δt	Time step
FEM	finite element Method
GFDM	generalized finite difference Method
EFG	element-free Galerkin
LPM	lumped-Parameter Models

KDC	Krylov deferred correction
MDOF	multi-degree-of-freedom
NFD	Newmark finite difference
FFDM	fast finite difference method
SOE	sum-of-exponentials
PGA	peak ground acceleration
RNA	rotor-nacelle-assembly

Chapter 1

Introduction

1.1 State of the art

1.1.1 The generalized finite difference method

The solution of differential equations by the Finite Difference Method (FDM) is a classic approach on regularly distributed points. It is a simple, intuitive and universal numerical method that deals directly with the differential form of the problem. However, disadvantages arise about the application of boundary conditions and the use of arbitrary grids, required for the solution on irregular geometries.

With the development of the finite element method (FEM), versatile and capable of to easily dealing with the problems mentioned above, the scientific interest of researchers in new procedures associated to the FDM gradually diminished. The FEM has been the dominant technique in computational mechanics in past decades, and has made significant contributions to the advancement of engineering and science.

However, the FEM can have a high computational cost in the meshing process. Consequently, in recent years, there has been a great interest in the meshless method in which neither elements nor mesh are needed. This increase is explained by Chen et al. [15]: "It discretizes the continuum body only with a set of nodal points and the approximation is constructed entirely in terms of nodes. The method is thus less susceptible to mesh distortion difficulties than the FE method. For a variety of

problems with extremely large deformation, moving boundary discontinuities, or in optimization problems where re-meshing may be required, meshless methods are very attractive. The method has the promise to provide an approach with more flexibility in the applications in engineering and science".

Meshless methods appeared from 1970 onwards in numerical simulations of astrophysics problems through the well-known smooth particle hydrodynamics (SPH) method [32]. Since then, many researchers have endeavored to increase the accuracy and the computational performance of such methods. Among the meshless methods, there are those used in the weak form [32, 62, 56, 55] and strong form [64, 54, 5, 52].

The generalized finite difference method (GFDM) is an extension of the classical finite difference method that is not restricted to a regular grid of points. The first contributions to the GFDM appeared in the 60s, for example, Forsythe and Wasow [23] introduced the possibility of using finite differences in irregular grids. In the following decade, Jensen [43] formulated the basis of the method and concluded that building the stars with the closest points may be inappropriate due to the emergence of ill-conditioned problems.

An important step in the development of the GFDM was the introduction of weighted moving least squares in the derivative approximations [84, 52]. Liszka and Orkisz [52] developed a robust algorithm, the FIDAM code, for the solution of linear and nonlinear problems in two-dimensional elliptic and three-dimensional parabolic equations. In particular, the problems of bar torsion, plane elasticity, deflections of plates and membranes, fluid flow, and temperature distribution were solved using the FIDAM code.

Later, Benito et al. [8] found an explicit formulation of the derivatives, obtained through the Cholesky decomposition, and analyzed the impact of different factors in the star concerning the accuracy of the derivatives using the second-order approximation. The three main conclusions were: (i) the results improve with the increase of points per star, but an exaggerated increase does not compensate for the computational effort required; (ii) the four quadrant criterion is the most suitable for the placement of points in the star; (iii) smooth weighting functions yield more accurate

results.

In Gavete et al. [28], the GFD method was compared with the element-free Galerkin (EFG) method. Both methods were tested for the Laplace equation with different domains and irregular mesh discretizations. For the tested cases, the GFD method appears to be more accurate compared to the EFG method with linear approximation. In Benito et al. [9], explicit equations for parabolic and hyperbolic problems were developed. The truncation errors and stability limit were also shown for those equations. An index of irregularity for a star and a discretization of points were proposed by Ureña et al. [77] to quantify the level of dispersion of the points in the domain. A study on the key parameters of the method in 3D applications was made by Ureña [74], where he highlighted that the number of points and the selection criteria per star should be analysed together.

The current scenario of the GFDM indicates a trend towards the use of higher-order approximations in second-order differential equations. One of the ways adopted was to consider additional correction terms [59, 61], where the correction was made by modifying only the right-hand sides of the algebraic equations; another way has been to use the idea of multipoint solution created by Collatz [18] that was extended to the GFDM [40, 41, 42], where additional degrees of freedom were added per star points.

1.1.2 Higher-order approximations

The use of second-order approximation on the GFDM has received much attention in the last two decades and has been successfully consolidated. Its applications vary among many types of problems, such as: inverse Cauchy problems in linear elasticity [51], natural frequency analysis of nanocomposite cylinders [38], hyperbolic nonlinear equations [76], transient heat flow in anisotropic composites [33], mathematical models of tumor growth [7], inverse heat conduction problem [39], discontinuous crack-faces [50], PDEs defined on manifolds [70], perfectly matched layer analysis of the problems in wave mechanics [49], mathematical model of the anti-plane elastic wave propagation in 2D solid phononic crystals [24], water-wave interactions with multiple-bottom-

seated-cylinder-array structures [25], fluid–structure coupling vibration response of thin plate structure [85], etc.

The fourth-order approximation may be used in the GFDM due to the higher-order operator inherent in the differential equation [30, 77] or to improve the approximation of derivatives with an order less than four. In addition, some authors propose the use of higher-order approximations to determine the errors in adaptive methods for solving second-order differential equations [10, 75, 12, 66]. However, higher-order approximations have been rarely used in the GFDM, due to two main reasons: the need to increase the minimum number of points per star, which often causes problems of ill-conditioning, and the high computational cost.

To avoid the use of the fourth-order approximation when the differential equation is of fourth-order, Orkisz [65] and Tseng and Gu [72] suggested decomposing the fourth-order operators by successive approximations using second-order operators. Nowadays, the power of computers allows using the fourth-order approximation with many more points per star than the minimum necessary, thus avoiding ill-conditioned stars. Therefore, some authors used the fourth-order approximation directly in the fourth-order differential equation: to solve plate bending problems in thin and thick elastic plates [77], in the dynamic analysis of beams and plates [30] or to solve the inverse biharmonic boundary value problems [21]. However, in higher-order approximations, a number of points per star has not been established as a reference, contrary to what happens in the second-order approximation, where eight points per star usually provides good results and is used as a reference value [8].

Regarding the use of higher-order operators in second-order differential equations, Benito et al. [10] concluded that reducing the error by using third-order operators does not justify the additional amount of calculations required. Milewski [59] presented a strategy in which higher-order terms were used as a correction to modify the right-hand side of the equation in GFDM, resulting in a two-step iterative procedure. Jaworska and Orkisz [41] introduced the idea of multipoint, which increases the order of approximation by introducing additional degrees of freedom at the star points.

1.1.3 Computational aspects

Meshless methods have been widely applied in various areas of engineering. However, in general applications, they still do not reach the computational efficiency in relation to traditional techniques in finite elements or finite volumes.

Python vector languages are widely used for scientific computing and there is significant interest in programming techniques in these languages for two reasons. The first is to make the code clear and compact, which provides rapid prototyping in research and industry. The second, of course, is to make the code fast enough for applications in realistic simulations. Additionally, vectorization is especially important in problems where the assembly of the arrays must be done multiple times, for instance, in nonlinear problems.

The use of parallel processing appears as an interesting and accessible workaround on multi-core computers. In Python, this procedure requires the use of the multiprocessing package, which performs simultaneous operations on the number of processors allocated to a task. Considering that in the GFDM each point is independent in the formation of stars and the calculation of the derivatives, then these steps can be parallelized among the available cores in a multiprocessor environment. In Ferreira and Ribeiro [22], the parallel process was used in the star formation with the MATLAB software.

Although the GFDM has demonstrated great ease of application in various complex engineering problems, not enough emphasis has been placed on computational efficiency. However, for a meshless technique to be widely applicable in industry, computational efficiency is a key factor.

1.1.4 Treatment of ill-conditioned stars

To avoid ill-conditioned stars or singularities, the main strategies used have been directed to the way of selecting the points in the star and the distribution of the points in the domain, considering for example Voronoi diagrams [65]. Another line of research focuses on the use of pseudo inverse matrices [4, 53].

The strategies used so far with schemes of any order to select the points in the star, as an alternative to the distance criterion, have been: the octant criterion proposed by Perrone and Kao [67], the quadrant criterion formulated by Liszka and Orkisz [52] and, as a complement to the quadrant criterion, Ferreira and Ribeiro [22] added an angular tolerance to avoid close points in the same quadrant or on the edge of two adjacent quadrants. All these selection criteria help to avoid ill-conditioned stars, but add computational cost compared to the minimum distance criterion. Hence, some authors suggest [77, 27, 78] the use of an index of the irregularity of the star to choose an appropriate selection criterion. However, there is also an extra computational cost in calculating these indices.

1.1.5 Different forms of discretization using the GFDM

The discretization of the domain when applying GFDM to solve a partial differential equation has been addressed in many ways. It can be discretized regularly whenever possible and irregularly only in regions where it cannot be done otherwise. In the latter case, there is usually some minimum distance criterion to prevent two points from being too close together.

It can also be discretized irregularly, either arbitrarily [19, 23, 43] or on the basis of some kind of structure such as, for example, using triangular elements (Delaunay triangulation) [60, 59, 83, 22], quadrilateral elements [16, 73], partitions into nodal subdomains (Voronoi tessellation) [65, 59] or Coatomèlec distribution of points [26].

All these discretizations have in common that the initial distribution of points is of approximately constant density throughout the domain. Liszka and Orkisz [52] developed a pre-process based on density functions that are defined by the user. However, there are not many papers where initial distributions adapted to the problem are applied, i.e., with a distribution of points that allows capturing the particularities of the problem or part of them. For example, in Benito et al. [13] a higher density of points near the interfaces is used, and in Ferreira and Ribeiro [22] a higher density of points near the boundary is used.

There are many papers where discretizations with different densities are used in

different regions of the domain, but these discretizations are the result of adaptive algorithms that refine the discretization in several steps depending on where the highest errors are.

1.1.6 Adaptive methods using the GFDM

There are several possible ways to improve the solution by the GFDM. One way is to increase the order of approximation of the derivatives, however, we do this by using fourth-order approximations since most authors use second-order approximations. Another way is increasing the number of points, this may be done by considering regular or irregular discretizations. In the last case, they may be generated using an h-adaptive approach.

The main advances in h-adaptive methods using the GFDM are described below.

Benito et al. [10] proposed an interesting error indicator based on a linear combination of the higher-order derivatives weighted by the coefficients of the stars. When adding points in the h-adaptive algorithm, special care is needed to avoid ill-conditioned stars. So, the authors proposed two key parameters for adding the points: at each central point a maximum of four points are added inside its star; and if the distance between the new point and any other point in the domain is less than a minimum distance, then the new point should not be added. The points are added halfway between the central point and the points of its star with greater values of error indicators.

An extension in 3D of the technique used in Benito et al.[10] can be seen in Ureña et al. [75], in this case a maximum of eight points was added to each star refined.

Benito et al. [12] developed an algorithm where the new points are added at the center of gravity of the triangles formed from the stars. In addition, the posteriori error indicator developed by Benito et al. [10] was compared with the posteriori error estimator created by Orkisz [65]. The results showed that, in general, the error estimator has a slight advantage concerning the error indicator. However, the disadvantage of the error estimator is that it needs to solve more than one system of equations.

A year later, Benito et al. [11] used the same idea with triangles but established

an additional criterion where points are not added in triangles with small areas.

A sophisticated version of adding points is provided by Ureña et al. [78]. Here, the algorithm allows not only adding, but also moving and deleting points. According to the authors: “ this movement algorithm helps to reduce the global error because of the reduction of the estimation of the error in the node itself as a consequence of improving the distribution of the nodes. Moreover, it produces a repulsion effect between nodes in areas with large values of the error, making spaces that can be occupied by new nodes if the addition algorithm was applied ”. At the end of the adaptive steps, the results show better accuracy with a decrease in the number of points compared to Benito et al. [11] and Ureña et al. [75] in 2D and 3D domains, respectively.

Another way chosen by some authors was to use an auxiliary background mesh to add points at each step of the adaptive procedure.

Liszka and Orkisz [52], Orkisz [65], and Orkisz and Milewski [66] developed an adaptive algorithm partitioning the domain into Voronoi polygons and Delaunay triangles. The candidate points are introduced using one level denser sieve (e.g., at the vertices of the elements created from Voronoi polygons). A residual error estimation was widely used in the adaptive mesh refinement technique. The residuals are examined in the candidate points only, being inserted if they exceed a maximum residual error.

The adaptive method proposed in Gavete et al. [29] is based on the use of quadtree and on the computations of the gradients, using the differences between the gradients in each quadtree as a simple error indicator. The main idea is that each quadrilateral cell selected for refinement is subdivided into four other quadrilaterals, one in each quadrant, consequently, 5 points are added per cell.

An extension in 3D of the algorithm quadtree was made by Gavete et al. [31], which was called octree algorithm. The cells are formed by cubes which in turn can be subdivided into eight other cubes, one in each octant, consequently, 19 points are added per cubic cell. Such a strategy can provoke an unnecessary increase of points in the same adaptive step. Then, the authors recommend that the refined cells should

be less than 15% of the number of cells in the domain in each step. Additionally, the authors clarify that the use of the octree structure in the adaptive refinement places points in a favorable way to avoid ill-conditioned stars.

1.1.7 Simplified models in the seismic response of wind turbines and the GFDM in dynamic responses

In seismically active areas the design of wind turbines must be verified for seismic load combinations. To simulate these seismic scenarios, the computational model should consider the aerodynamics of the rotor, the flexibility of the tower and soil, transient operational phases, and the interrelation of all these aspects. At the same time, the complexity should be reduced to avoid enormous computational costs, especially when the soil-structure interaction is considered. Therefore, it is natural that simplified models, either analytical or numerical, replace, for example, three-dimensional complex models.

Zhao and Maisser [87] and Jin et al. [45] used the dynamic theory of multi-body system in the analysis of the dynamic response of the tower under earthquake action. The soil-structure effect was considered with the presence of spring and dampers. Malaeke and Moeenfarid [57] analysed the dynamic response of a non-uniform cantilever beam carrying an eccentric tip mass. The model is governed by two non-linear ordinary differential equations, which in turn are solved analytically using the multiple time scale perturbation technique. Taddei et al. [71] presented a model based on the LPM (Lumped-Parameter Models) method for the analysis of the effects of soil-structure interaction on the seismic behavior of wind turbines. Recently [14], a multi-degree-of-freedom (MDOF) system was used to describe a wind turbine system with the consideration of the soil-structure interaction. Due to the increasing demand for offshore applications, the seismic response in this type of structure has also been studied [58, 44, 86, 2].

Regarding the application of the GFDM in partial differential equations involving the time variable, the most common way [9, 30, 79, 13] is to use the GFDM to

approximate the spatial derivatives together with classical finite-difference operators to approximate the time derivatives. In this case, the solution at a point of time level $k + 1$ is expressed explicitly in terms of the known solution at time level k . The explicit finite difference method is conditionally stable, consequently, the required step size is generally quite small, and the amount of computational effort required to obtain the solution of some problems can be large.

Other combinations also were proposed. Hosseini [37] proposed a hybrid mesh-free method based on GFDM and Newmark Finite Difference (NFD) methods to calculate the velocity of elastic wave propagation in functionally graded materials (FGMs). Gu et al. [34] developed a combined Krylov deferred correction (KDC)-GFDM scheme for long-time dynamic simulations in thermoelasticity problems. Recently, Wang and Sun [82] combined the GFDM with a fast finite difference method (FFDM) based on sum-of-exponentials (SOE) approximation in the solution of the Fractional advection-diffusion equation.

In Gavete et al. [30], the GFDM was applied in a simplified model of Euler-Bernoulli beam vibration with constant section and fixed, supported, and free boundary conditions. The von Neumann stability criterion was applied to ensure the stability of the solution in the time variable.

Considering the geometric properties of a wind turbine, it is easily observable that such a structure can be approximately described by a model of Euler-Bernoulli transverse beam vibration. Within these geometric properties, we can highlight the variable cross-section throughout the tower, rotary inertia and mass at the top of the tower, and springs at the base describing the soil-structure interaction. Until now, there is no application of the GFDM in the literature for the solution of the dynamic response of this type of structure.

1.2 Objectives

The main objectives of this thesis are

- provide a number of points per star as a reference in third- and fourth-order

approximations;

- reduce the computational cost in different stages of the method;
- get a strategy to detect and correct ill-conditioned stars;
- designing a strategy to generate a discretization adapted to the problem in a general way;
- obtain a 3D adaptive algorithm using fourth-order approximation;
- obtain the seismic response of wind turbines using the GFDM coupled with the Newmark method.

Chapter 2

The generalized finite difference method

Let $D \subset \mathbb{R}^\Phi$, $\Phi = 1, 2, 3$, be a domain with interior Ω and boundary Γ . Let us consider $\mathbf{x} = x$ for $\Phi = 1$, $\mathbf{x} = (x, y)$ for $\Phi = 2$, and $\mathbf{x} = (x, y, z)$ for $\Phi = 3$.

Let M be a discretization of D with N points. For the ease of notation and, without loss of generality, let us consider a set of m different points of M , say $V = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, and denote a generic point of $M - V$ as \mathbf{x}_0 , that is, $\mathbf{x}_0 \in (M - V)$. The pair (\mathbf{x}_0, V) is called a star with central point \mathbf{x}_0 , and the elements of V are called the points of the star. We assume that the points of V are chosen in such way that each line joining the central point and any point of V is entirely contained in D .

Consider any function $U(\mathbf{x})$ determined on the domain D with U being sufficiently derivable in \mathbf{x}_0 . For each $\mathbf{x}_i \in V$, $i = 1, 2, \dots, m$, we consider the Taylor expansion of $U(\mathbf{x})$ centered at \mathbf{x}_0 where we assume that two higher-order mixed partial derivatives that involve the same number of differentiations in each variable are equal (Clairaut-Schwarz Theorem). If we truncate the derivatives after second-, third- or fourth-order we obtain respectively approximations of such orders. We consider here the truncation of the terms after fourth-order. Then, we can write the residuals as

$$\mathbf{r} = \Delta \mathbf{u} - \mathbf{P} \mathbf{D} \mathbf{u}, \tag{2.1}$$

being

$$\mathbf{P} = \begin{cases} \left[\begin{array}{cccc} h_1 & \frac{h_1^2}{2} & \frac{h_1^3}{6} & \frac{h_1^4}{24} \\ h_2 & \frac{h_2^2}{2} & \frac{h_2^3}{6} & \frac{h_2^4}{24} \\ \vdots & \vdots & \vdots & \vdots \\ h_m & \frac{h_m^2}{2} & \frac{h_m^3}{6} & \frac{h_m^4}{24} \end{array} \right], & \text{if } \Phi = 1 \\ \left[\begin{array}{cccccc} h_1 & k_1 & \frac{h_1^2}{2} & \frac{k_1^2}{2} & h_1 k_1 & \dots & \frac{h_1 k_1^3}{6} & \frac{k_1^4}{24} \\ h_2 & k_2 & \frac{h_2^2}{2} & \frac{k_2^2}{2} & h_2 k_2 & \dots & \frac{h_2 k_2^3}{6} & \frac{k_2^4}{24} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h_m & k_m & \frac{h_m^2}{2} & \frac{k_m^2}{2} & h_m k_m & \dots & \frac{h_m k_m^3}{6} & \frac{k_m^4}{24} \end{array} \right], & \text{if } \Phi = 2 \quad (2.2) \\ \left[\begin{array}{ccccccc} h_1 & k_1 & l_1 & \frac{h_1^2}{2} & \frac{k_1^2}{2} & \frac{l_1^2}{2} & \dots & \frac{h_1^4}{24} & \frac{k_1^4}{24} & \frac{l_1^4}{24} \\ h_2 & k_2 & l_2 & \frac{h_2^2}{2} & \frac{k_2^2}{2} & \frac{l_2^2}{2} & \dots & \frac{h_2^4}{24} & \frac{k_2^4}{24} & \frac{l_2^4}{24} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ h_m & k_m & l_m & \frac{h_m^2}{2} & \frac{k_m^2}{2} & \frac{l_m^2}{2} & \dots & \frac{h_m^4}{24} & \frac{k_m^4}{24} & \frac{l_m^4}{24} \end{array} \right], & \text{if } \Phi = 3 \end{cases}$$

$$\mathbf{D}_u = \begin{cases} \left(\frac{du_0}{dx}, \frac{d^2u_0}{dx^2}, \frac{d^3u_0}{dx^3}, \frac{d^4u_0}{dx^4} \right)^T & \text{if } \Phi = 1 \\ \left(\frac{\partial u_0}{\partial x}, \frac{\partial u_0}{\partial y}, \frac{\partial^2 u_0}{\partial x^2}, \frac{\partial^2 u_0}{\partial y^2}, \frac{\partial^2 u_0}{\partial x \partial y}, \dots, \frac{\partial^4 u_0}{\partial x \partial y^3}, \frac{\partial^4 u_0}{\partial y^4} \right)^T & \text{if } \Phi = 2 \quad (2.3) \\ \left(\frac{\partial u_0}{\partial x}, \frac{\partial u_0}{\partial y}, \frac{\partial u_0}{\partial z}, \frac{\partial^2 u_0}{\partial x^2}, \frac{\partial^2 u_0}{\partial y^2}, \frac{\partial^2 u_0}{\partial z^2}, \dots, \frac{\partial^4 u_0}{\partial x^4}, \frac{\partial^4 u_0}{\partial y^4}, \frac{\partial^4 u_0}{\partial z^4} \right)^T & \text{if } \Phi = 3 \end{cases}$$

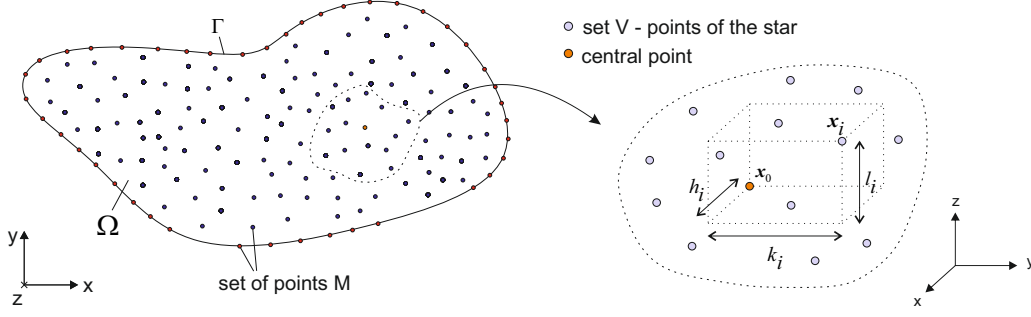


Figure 2-1: Discretization of the domain and the star formation in 3D (h_i, k_i and l_i are the relative distances of \mathbf{x}_i to \mathbf{x}_0)

$$\Delta \mathbf{u} = \begin{bmatrix} u_1 - u_0 \\ u_2 - u_0 \\ \vdots \\ u_m - u_0 \end{bmatrix}, \quad (2.4)$$

where $u_i, i = 0, 1, \dots, m$, denote approximate values of $U_i = U(\mathbf{x}_i)$, $\mathbf{D}_{\mathbf{u}}$ corresponds to the approximate derivatives at the central point and $h_i = x_i - x_0$, $k_i = y_i - y_0$ and $l_i = z_i - z_0$ are the relative coordinates of the points of the star (see Figure 2-1).

To minimize the residuals we consider the method of least squares applied to the weighted residual given by $\mathbf{r}^T \mathbf{W}^2 \mathbf{r}$, where

$$\mathbf{W} = \begin{pmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_m \end{pmatrix} \quad (2.5)$$

is a diagonal matrix of weights. Solving the system

$$\frac{\partial(\mathbf{r}^T \mathbf{W}^2 \mathbf{r})}{\partial \mathbf{D}_{\mathbf{u}}} = 0, \quad (2.6)$$

we obtain the following approximate values for the derivatives

$$\mathbf{D}_{\mathbf{u}} = (\mathbf{P}^T \mathbf{W}^2 \mathbf{P})^{-1} \mathbf{P}^T \mathbf{W}^2 \Delta \mathbf{u}. \quad (2.7)$$

The prove of (2.7) is shown as follow

$$\begin{aligned}
\mathbf{r}^T \mathbf{W}^2 \mathbf{r} &= (\Delta \mathbf{u} - \mathbf{P} \mathbf{D} \mathbf{u})^T \mathbf{W}^2 (\Delta \mathbf{u} - \mathbf{P} \mathbf{D} \mathbf{u}) \\
&= \Delta \mathbf{u}^T \mathbf{W}^2 \Delta \mathbf{u} - \Delta \mathbf{u}^T \mathbf{W}^2 \mathbf{P} \mathbf{D} \mathbf{u} - (\mathbf{P} \mathbf{D} \mathbf{u})^T \mathbf{W}^2 \Delta \mathbf{u} + (\mathbf{P} \mathbf{D} \mathbf{u})^T \mathbf{W}^2 \mathbf{P} \mathbf{D} \mathbf{u} \quad (2.8) \\
&= \Delta \mathbf{u}^T \mathbf{W}^2 \Delta \mathbf{u} - \mathbf{D} \mathbf{u}^T (\Delta \mathbf{u}^T \mathbf{W}^2 \mathbf{P})^T - \mathbf{D} \mathbf{u}^T \mathbf{P}^T \mathbf{W}^2 \Delta \mathbf{u} + \mathbf{D} \mathbf{u}^T \mathbf{P}^T \mathbf{W}^2 \mathbf{P} \mathbf{D} \mathbf{u}.
\end{aligned}$$

Substituting the least line of (2.8) in (2.6) we get

$$\frac{\partial(\mathbf{r}^T \mathbf{W}^2 \mathbf{r})}{\partial \mathbf{D} \mathbf{u}} = -\mathbf{P}^T \mathbf{W}^2 \Delta \mathbf{u} - \mathbf{P}^T \mathbf{W}^2 \Delta \mathbf{u} + 2\mathbf{P}^T \mathbf{W}^2 \mathbf{P} \mathbf{D} \mathbf{u} = 0, \quad (2.9)$$

and therefore

$$-2\mathbf{P}^T \mathbf{W}^2 \Delta \mathbf{u} + 2\mathbf{P}^T \mathbf{W}^2 \mathbf{P} \mathbf{D} \mathbf{u} = 0 \Rightarrow \mathbf{D} \mathbf{u} = (\mathbf{P}^T \mathbf{W}^2 \mathbf{P})^{-1} \mathbf{P}^T \mathbf{W}^2 \Delta \mathbf{u}. \quad (2.10)$$

Note that if \mathbf{P} is a square matrix, then the derivatives do not depend on the weight matrix.

The equation (2.7) can be written in the compact form:

$$\mathbf{D} \mathbf{u} = \mathbf{B} \Delta \mathbf{u}, \quad (2.11)$$

where the matrix \mathbf{B} is given by

$$\mathbf{B} = \mathbf{A}^{-1} \mathbf{P}^T \mathbf{W}^2, \quad (2.12)$$

with

$$\mathbf{A} = \mathbf{P}^T \mathbf{W}^2 \mathbf{P}. \quad (2.13)$$

In the least squares minimization the matrix \mathbf{A} is invertible if and only if the rows of the matrix \mathbf{P} contain a base of \mathbb{R}^4 if $\Phi = 1$, \mathbb{R}^{14} if $\Phi = 2$ or \mathbb{R}^{34} if $\Phi = 3$ (see [79]).

Denoting $\mathbf{u} = (u_0, u_1, \dots, u_m)^T$ and

$$\bar{\mathbf{B}} = \begin{bmatrix} -\sum_{i=1}^m b_{1,i} & b_{1,1} & b_{1,2} & \dots & b_{1,m} \\ -\sum_{i=1}^m b_{2,i} & b_{2,1} & b_{2,2} & \dots & b_{2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\sum_{i=1}^m b_{p,i} & b_{p,1} & b_{p,2} & \dots & b_{p,m} \end{bmatrix}, \quad (2.14)$$

where the elements $b_{i,j}$ are those of the matrix \mathbf{B} and p is the number of columns in \mathbf{P} , we get the formula

$$\mathbf{D}_u = \mathbf{B}\Delta u = \bar{\mathbf{B}}\mathbf{u} = \begin{bmatrix} -u_0 \sum_{i=1}^m b_{1,i} + \sum_{i=1}^m b_{1,i}u_i \\ -u_0 \sum_{i=1}^m b_{2,i} + \sum_{i=1}^m b_{2,i}u_i \\ \vdots \\ -u_0 \sum_{i=1}^m b_{p,i} + \sum_{i=1}^m b_{p,i}u_i \end{bmatrix} = \begin{bmatrix} -u_0 \sum_{i=1}^m b_{1,i} \\ -u_0 \sum_{i=1}^m b_{2,i} \\ \vdots \\ -u_0 \sum_{i=1}^m b_{p,i} \end{bmatrix} + \begin{bmatrix} \sum_{i=1}^m b_{1,i}u_i \\ \sum_{i=1}^m b_{2,i}u_i \\ \vdots \\ \sum_{i=1}^m b_{p,i}u_i \end{bmatrix}. \quad (2.15)$$

Note that the equation in (2.15) does not depend on the differential equation, but it depends on

- i) the number of points of the star, m ,
- ii) the weighting function used (if $m > p$),
- iii) the geometry of the star.

Theoretical issues as consistency, order, stability, and convergence of the GFDM, have been studied in previous works [30, 27, 9].

2.1 Solution of a Dirichlet boundary value problem by the GFDM

Consider a boundary value problem of Dirichlet type given by

$$\mathcal{L}(U(\mathbf{x})) = f(\mathbf{x}), \mathbf{x} \in \Omega \quad (2.16)$$

$$U(\mathbf{x}) = g(\mathbf{x}), \mathbf{x} \in \Gamma \quad (2.17)$$

where \mathcal{L} is a second-order linear differential operator with constant coefficients, and f and g are known real functions, where we assume enough regularity in order that the above problem has a unique solution. Our goal is to obtain approximate values of U at the points of $M \cap \Omega$.

We consider $a_0 \in \mathbb{R}$ and a vector \mathbf{a} with the same size as $\mathbf{D}_{\mathbf{u}}$, where a_0 is the coefficient of U in (2.16) and \mathbf{a} contains the coefficients of the partial derivatives of the linear differential operator in (2.16) placed according to the corresponding elements of $\mathbf{D}_{\mathbf{u}}$:

$$\mathbf{a} = (a_1 \ a_2 \ \dots \ a_p). \quad (2.18)$$

Note that in (2.18) the elements that multiply the third and fourth-order partial derivatives are zero. Then, we can substitute the values of the approximate derivatives obtained in (2.15) into (2.16) to get

$$a_0 u_0 + \mathbf{a} \mathbf{D}_{\mathbf{u}} = f(\mathbf{x}_0) \implies a_0 u_0 + \mathbf{a} \bar{\mathbf{B}} \mathbf{u} = f(\mathbf{x}_0). \quad (2.19)$$

Note that the equation in (2.19) was obtained for a generic interior point of M . If we evaluate (2.19) at each interior point M , we get a system of linear equations

$$\mathbb{K} \bar{\mathbf{u}} = \mathbf{F} \quad (2.20)$$

where \mathbb{K} is a matrix of dimension $\text{NI} \times \text{NI}$, with NI the total number of interior points of M , and where each row has at most $m + 1$ terms different from zero, the vector \mathbf{F}

contains the values of f at each interior point of M and $\tilde{\mathbf{u}}$ is the vector of approximate values of U at the points of M . When the boundary conditions in (2.17) are evaluated, we get a system of linear equations with N_I equations and N_I unknowns:

$$\bar{\mathbb{K}}\bar{\mathbf{u}} = \bar{\mathbf{F}}, \quad (2.21)$$

where $\bar{\mathbb{K}}$ is the square matrix formed by \mathbb{K} without the columns corresponding to the boundary values, the vector $\bar{\mathbf{F}}$ is obtained through the vector \mathbf{F} and the known boundary values, and $\bar{\mathbf{u}}$ is the vector of approximate values of U at the interior points of M .

Most of the applications in this thesis involve Dirichlet problems. However, the application of the method in chapter 7 involves equations with initial conditions and Neumann boundary conditions. In this case, the procedure will be explained in chapter 7.

2.2 Influence of the number of points per star with higher-order approximations

We analyzed the influence of the number of points per star on higher-order approximations for 2D and 3D in sections (2.2.1) and (2.2.2), respectively.

To do this, we consider some discretized domains in 2D and 3D, as follows:

- The interior of the domain is discretized regularly taking a stepsize (h, h) in 2D and (h, h, h) in 3D. The points close to the boundary already present enough irregularity but, despite this, we also carried out examples allowing small random displacements of the interior points with respect to their position of regularity, with the intention of making the discretization even more irregular.

- Interior points that are at a distance less than $h/2$ from the boundary are not placed to avoid ill-conditioned stars due to points that are too close.

Furthermore, we also consider two domains discretized in an absolutely random way.

In all cases, and for each star, we used the weighting function $w_i = \|\mathbf{x}_i - \mathbf{x}_0\|_2^{-4}$, $i = 1, 2, \dots, m$, and the distance criterion to form the stars.

2.2.1 Approximations in 2D

We consider four domains, D1, ID1, D2, and D3, given by

$$D1 = ID1 = \{(x, y) \in \mathbb{R}^2 | 0 \leq x, y \leq 1\}$$

$$D2 = \{(x, y) \in \mathbb{R}^2 | 0.5^2 \leq x^2 + y^2 \leq 1.5^2\}$$

$$D3 = \{(x, y) \in \mathbb{R}^2 | 0 \leq x \leq 3, 0 \leq y \leq 5\} - (C1 \cup C2 \cup C3),$$

with $C1 = \{(x, y) \in \mathbb{R}^2 | (x - 0.75)^2 + (y - 4)^2 < 0.25^2\}$, $C2 = \{(x, y) \in \mathbb{R}^2 | (x - 0.8)^2 + (y - 1)^2 < 0.4^2\}$ and $C3 = \{(x, y) \in \mathbb{R}^2 | (x - 2)^2 + (y - 2)^2 < 0.6^2\}$, and discretizations: 188 points for D1 and ID1, 799 points for D2, and 934 points for D3 (see Figure 2-2).

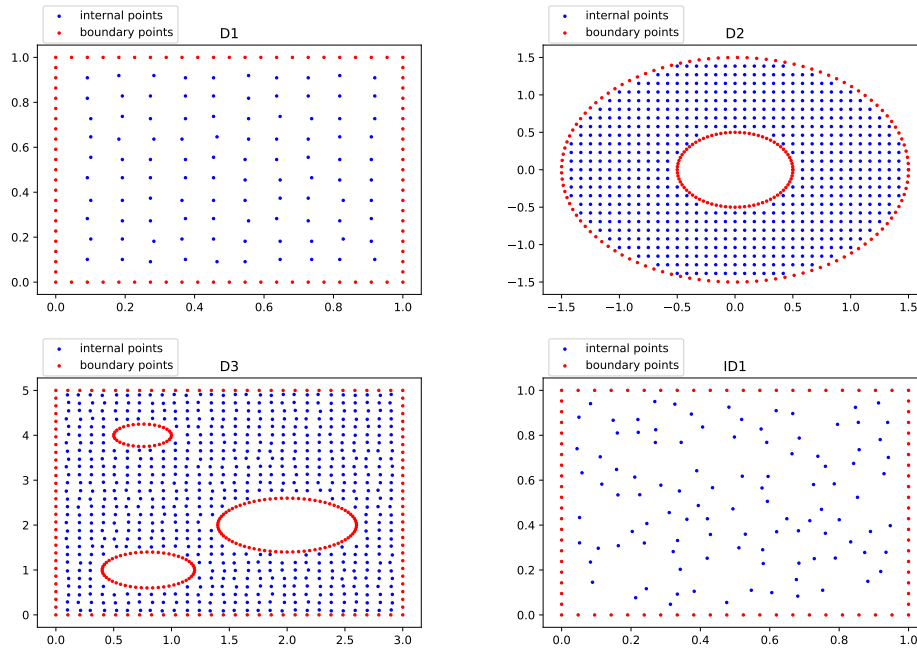


Figure 2-2: The domains and discretizations used in the 2D examples.

We consider three boundary value problems given by the following partial differ-

ential equations

$$2\frac{\partial^2 U}{\partial x^2} - \frac{\partial^2 U}{\partial y^2} + 2\frac{\partial U}{\partial x} = (x+2)^2 \cos(y), \quad (2.22)$$

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0, \quad (2.23)$$

$$\frac{\partial^2 U}{\partial x^2} + 2\frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial x \partial y} + \frac{\partial U}{\partial x} - \frac{\partial U}{\partial y} = 0, \quad (2.24)$$

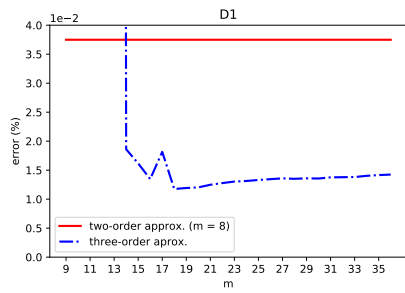
with exact solutions $U(x, y) = x^2 \cos(y)$, $U(x, y) = \ln(x^2 + y^2)$ and $U(x, y) = e^x \sin(y)$, respectively. The corresponding boundary values are obtained from the exact solutions.

We solve the equation (2.22) in the domains D1 and ID1, the equation (2.23) in D2, and the equation (2.24) in D3.

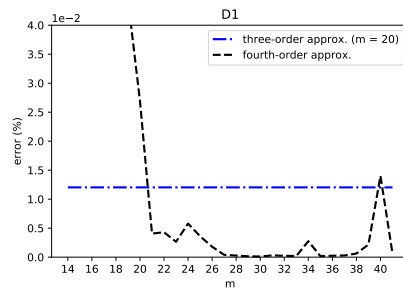
For the third-order approximation, we calculate the global error by increasing the number of points in the star starting with the minimum number necessary, which is 9 points, and we compare those errors with the error caused by using stars with 8 points for the second-order approximation. Plots (a), (c), (e) and (g) in Figure 2-3 show that 20 points per star can be an appropriate reference.

For the fourth-order approximation, we calculate the global error by increasing the number of points in the star starting with the minimum number necessary, which is 14 points, and we compare those errors with the error caused by using stars with 20 points for the third-order approximation. Plots (b), (d), (f) and (h) in Figure 2-3 show that 30 points per star can be an appropriate reference.

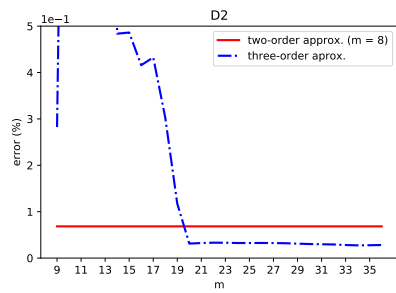
In both cases we have chosen the number of reference points as the first value from which the variations between the errors for consecutive values are stable for the first time. Therefore, this reference value serves as a starting point, but this does not mean that large errors cannot occur with a larger number of points, m , per star.



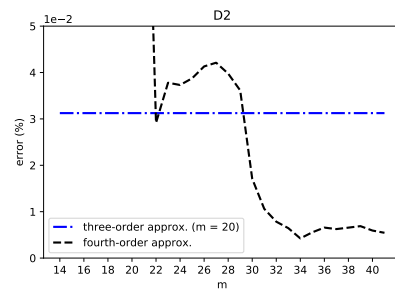
(a)



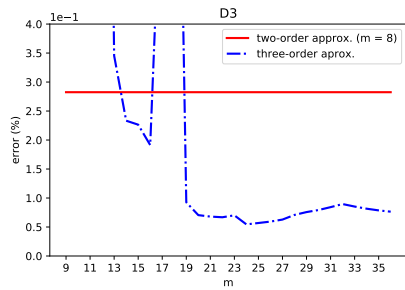
(b)



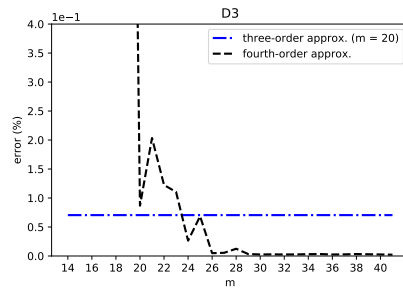
(c)



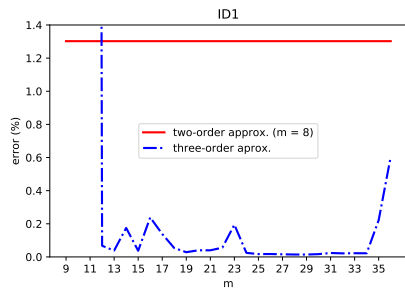
(d)



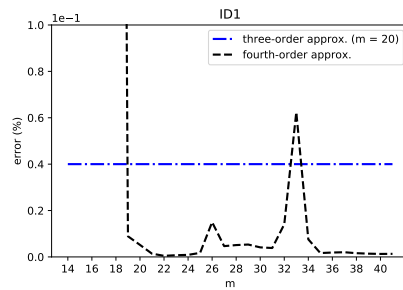
(e)



(f)



(g)



(h)

Figure 2-3: Numerical errors for each approximation order in 2D

2.2.2 Approximations in 3D

We consider now the domains D4, ID4, D5 and D6, given by

$$D4 = ID4 = \{(x, y, z) \in \mathbb{R}^3 | 0 \leq x, y, z \leq 1\}$$

$$D5 = \{(x, y, z) \in \mathbb{R}^3 | x^2 + y^2 + z^2 \leq 1\}$$

$$D6 = \{(x, y, z) \in \mathbb{R}^3 | (1 - \sqrt{x^2 + y^2})^2 + z^2 \leq 0.5^2\},$$

, taking discretizations with 233 points for D4 and ID4, 604 points for D5, and 4161 points for D6 (see Figure 2-4).

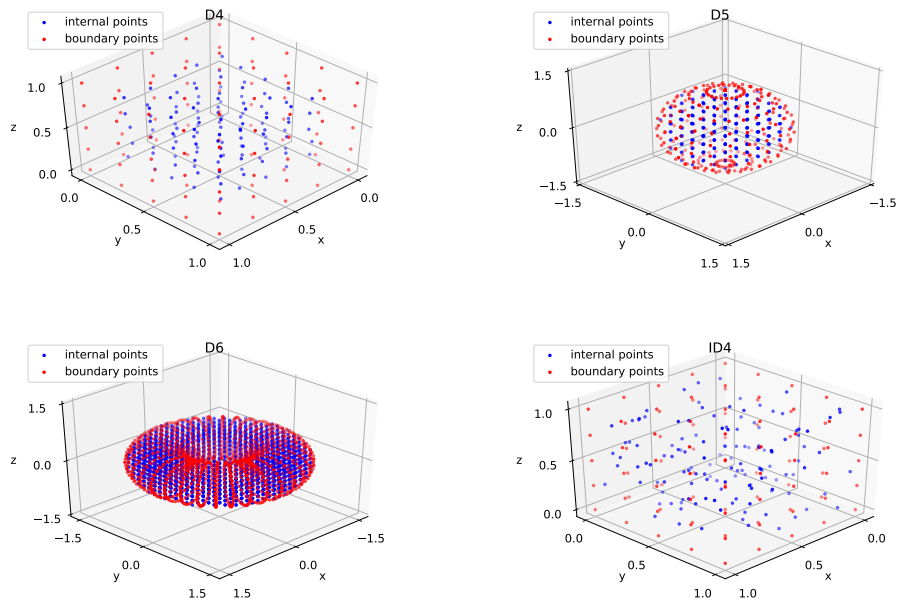


Figure 2-4: The domains and discretizations used in the 3D examples.

We consider three boundary value problems given by the partial differential equations

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial U}{\partial z} = (1 - z)(\sin(x) + \sin(y)) \quad (2.25)$$

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0 \quad (2.26)$$

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 2(y^2 z^2 + x^2 z^2 + x^2 y^2), \quad (2.27)$$

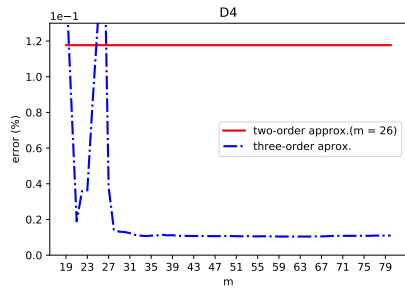
with exact solutions $U(x, y, z) = z(\sin(x) + \sin(y))$, $U(x, y, z) = e^x \sin(y) + e^y \sin(z) + e^z \sin(x)$, and $U(x, y, z) = x^2 y^2 z^2$, respectively. The boundary values are obtained from the exact solutions.

We solve the equation (2.25) in the domains D4 and ID4, the equation (2.26) in D5 and the equation (2.27) in D6.

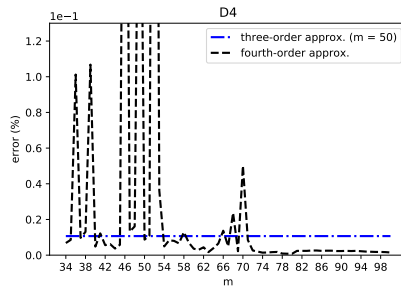
For the third-order approximation, we calculate the global error by increasing the number of points per star starting with the minimum number necessary, which is 19 points, and we compare those errors with the error caused by using stars with 26 points for the second-order approximation. Plots (a), (c), (e) and (g) in Figure 2-5 show that 50 points per star can be an appropriate reference.

For the fourth-order approximation, we calculate the global error by increasing the number of points per star starting with the minimum number necessary, which is 34 points, and we compare those errors with the error caused by using stars with 50 points for the third-order approximation. Plots (b), (d), (f) and (h) in Figure 2-5 show that 90 points per star can be an appropriate reference.

In both cases we have chosen the number of reference points as the first value from which the variations between the errors for consecutive values are stable for the first time. Therefore, this reference value serves as a starting point, but this does not mean that large errors can not occur with a larger number of points.



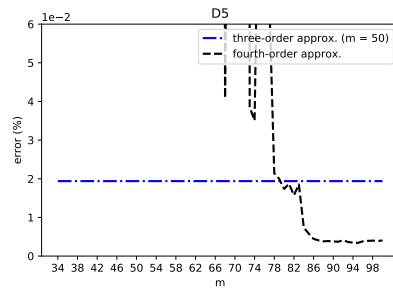
(a)



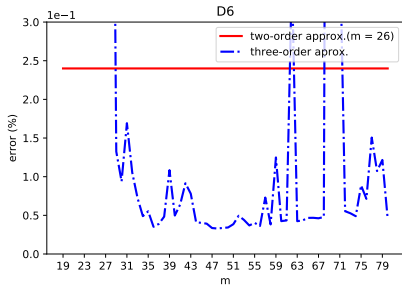
(b)



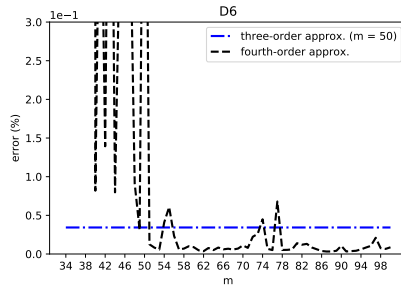
(c)



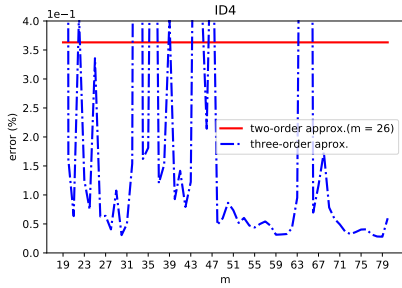
(d)



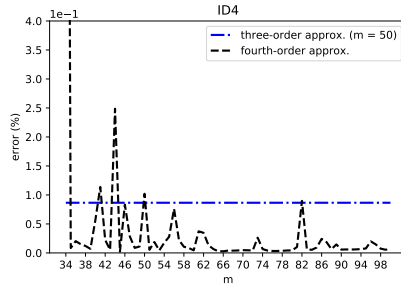
(e)



(f)



(g)



(h)

Figure 2-5: Numerical errors for each order of approximation in 3D applications

Chapter 3

Computational aspects

In recent years there has been a growing interest in the GFDM. While current research efforts are primarily aimed at applying these techniques to the analysis of various complex engineering problems, not enough emphasis is placed on computational efficiency. Considering the increase in computational cost when using higher-order approximations, these aspects cited above become even more relevant.

The codes to perform the numerical experiments have been written in Python. We have performed all the simulations of this thesis with an Intel i7-8750H processor (with six cores and maximum frequency 2.2GHz).

The \mathbb{K} matrix is a sparse matrix and the programming routine follows the recommended good programming practices for sparse assembly matrices [20]. To assemble the global \mathbb{K} matrix, we calculate and store all the point contributions first and then they are used to generate the sparse \mathbb{K} matrix. The main idea is to create four global 1D-arrays \mathbf{B}_g , \mathbf{I}_g , \mathbf{J}_g and \mathbf{a}_g (the subindex g denotes that they are global variables). The vector \mathbf{B}_g is formed by the contribution of each local matrix $\bar{\mathbf{B}}$ (2.14) stored row-wise

$$\mathbf{B}_g = \left(\overbrace{- \sum_{i=1}^m b_{1,i} \ b_{1,1} \dots \ b_{1,m} \ - \sum_{i=1}^m b_{2,i} \ b_{2,1} \dots \ b_{2,m} \dots \ - \sum_{i=1}^m b_{p,i} \ b_{p,1} \dots \ b_{p,m}}^{\text{NI times}} \right), (3.1)$$

a pseudocode for assembling the vector \mathbf{B}_g is shown in Algorithm 1.

Algorithm 1 A pseudocode to assemble the vector \mathbf{B}_g in Python

Require: import numpy as np
Initialize the global vector \mathbf{B}_g
Calculate and store the stars
for $i = 1 \dots \text{NI}$ **do** ▷ loop to each central point
 Assemble the matrices \mathbf{P} and \mathbf{W} of one star ▷ equations (2.2) and (2.5)
 $\mathbf{A} = \text{np.matmul}(\text{np.matmul}(\mathbf{P.T}, \mathbf{W}^{**2}), \mathbf{P})$ ▷ equation (2.13)
 $\mathbf{B} = \text{np.matmul}(\text{np.matmul}(\text{inv}(\mathbf{A}), \mathbf{P.T}), \mathbf{W}^{**2})$ ▷ equation (2.12)
 $\bar{\mathbf{B}} = \text{np.hstack}((-\text{np.sum}(\mathbf{B}, 1, \text{keepdims}=\text{True}), \mathbf{B}))$ ▷ equation (2.14)
 $\mathbf{B}_g \leftarrow \bar{\mathbf{B}}.\text{flatten}()$ ▷ collapse the matrix $\bar{\mathbf{B}}$ in one direction and store it in \mathbf{B}_g
end for

\mathbf{I}_g denotes the global row indices associated with the elements stored in $\bar{\mathbf{B}}$ and \mathbf{J}_g refers to the global column indices associated with the elements stored in $\bar{\mathbf{B}}$.

Finally,

$$\mathbf{a}_g = \left(\overbrace{\underbrace{a_1 a_1 \dots a_1}_{m+1} \underbrace{a_2 a_2 \dots a_2}_{m+1} \underbrace{a_3 a_3 \dots a_3}_{m+1} \dots \underbrace{a_p a_p \dots a_p}_{m+1}}^{\text{NI times}} \right). \quad (3.2)$$

The number of points per star, m , used in (3.1) and (3.2) has local numbering. Note that this value can change of one star to another.

The global \mathbb{K} matrix is assembled in the form

$$\mathbb{K} = \text{sparse.coo_matrix}((\mathbf{a}_g \circ \mathbf{B}_g, (\mathbf{I}_g, \mathbf{J}_g))) + \text{diag}(a_0, a_0, \dots, a_0)[46],$$

where \circ is the Hadamard product. Notice that in the current procedure used, the instruction $\mathbb{K}(i, j) = \dots$ does not exist (i and j are scalars), this statement is quite slow when \mathbb{K} is large and sparse. This leads us to an improvement in the execution time of the code.

Note that the size of the vectors \mathbf{I}_g , \mathbf{J}_g , \mathbf{B}_g and \mathbf{a}_g is given by $p \cdot (m + 1) \cdot \text{NI}$, provided that all the stars have the same number of points, m .

3.1 Parallel process in the GDFM

As a consequence of the independence for each interior point of the calculations for the formation of the stars and of the calculations of the derivatives, we can parallelize the GDFM and distribute the work among the available cores on a multiprocessor environment. This task is performed as follows:

1. Parallelization of the stars: the calculation of the points of the star of each interior point is an independent task and, therefore, the work can be distributed among the available cores.
2. Parallelization of the derivatives: this is similar to the parallel process in the construction of the global vectors \mathbf{B}_g , \mathbf{I}_g , \mathbf{J}_g and \mathbf{a}_g , which is distributed by multiple independent tasks. These global vectors are arrays allocated in shared memory using the multiprocessing.Array [68] in Python.

Note that the formation of stars and the calculation of the derivatives do not depend on the differential equation.

3.2 A vectorized algorithm in the GDFM

In the current section, we will present the vectorized version of the algorithm 1 where the loop at each central point is removed to compute the derivatives. The main idea consists in transforming the 2D arrays \mathbf{P} and \mathbf{W} into 3D arrays, where the third index is used to store the \mathbf{P} and \mathbf{W} arrays of each star of the domain. Therefore, we

have

$$\begin{aligned}
\mathbf{h}_1 &= (h_1^1 \ h_1^2 \ \dots \ h_1^{\text{NI}}) \\
\mathbf{k}_1 &= (k_1^1 \ k_1^2 \ \dots \ k_1^{\text{NI}}) \\
\mathbf{l}_1 &= (l_1^1 \ l_1^2 \ \dots \ l_1^{\text{NI}}) \\
&\vdots \\
\mathbf{h}_m &= (h_m^1 \ h_m^2 \ \dots \ h_m^{\text{NI}}) \\
\mathbf{k}_m &= (k_m^1 \ k_m^2 \ \dots \ k_m^{\text{NI}}) \\
\mathbf{l}_m &= (l_m^1 \ l_m^2 \ \dots \ l_m^{\text{NI}}),
\end{aligned} \tag{3.3}$$

where the super-indices are the global numbering of the central points. For instance: h_5^8, k_5^8 and l_5^8 are the relative distances from the eighth central point to the fifth point of its star. The 3D arrays $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{W}}$ are obtained from (3.3), as shown in Algorithm 2. We recall that m and p are the number of points per star and the number of columns in \mathbf{P} , respectively.

Algorithm 2 A vectorized pseudocode to assemble the 3D arrays $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{W}}$ in Python

```

Initialize the global 3D arrays  $\tilde{\mathbf{P}}$  and  $\tilde{\mathbf{W}}$ 
for  $i = 1 \dots m$  do
     $\tilde{\mathbf{P}}[:,i,1] = \mathbf{h}_i$ 
     $\tilde{\mathbf{P}}[:,i,2] = \mathbf{k}_i$ 
     $\tilde{\mathbf{P}}[:,i,3] = \mathbf{l}_i$ 
     $\tilde{\mathbf{P}}[:,i,4] = \mathbf{h}_i ** 2 / 2$ 
     $\tilde{\mathbf{P}}[:,i,5] = \mathbf{k}_i ** 2 / 2$ 
     $\tilde{\mathbf{P}}[:,i,6] = \mathbf{l}_i ** 2 / 2$ 
     $\vdots$ 
     $\tilde{\mathbf{P}}[:,i,p] = \dots$ 
     $\tilde{\mathbf{W}}[:,i,i] = \text{funWeigh}(\mathbf{h}_i, \mathbf{k}_i, \mathbf{l}_i)$   $\triangleright$  funWeigh is the weighting function chosen
end for

```

Note that in Python, the third dimension is placed at the first index of a 3D array. In our case, the third dimension refers to each star of the domain. For this reason, the symbol ':' sweeps all the stars of the domain.

Also note that the inserted loop depends on the number of points per star, and has a negligible computational cost compared to the loop of algorithm 1 (of course, we are

considering a problem with a large number of points in M , otherwise vectorization would not be necessary).

The 3D array $\tilde{\mathbf{B}}$ stores the array \mathbf{B} of all the stars, it is calculated with only three multiplications, as can be seen in Algorithm 3. The command `matmul` is capable of performing multiplications between 3D arrays. In Algorithms 2 and 3, the broadcasting rules in Python facilitate operations with 3D arrays.

Algorithm 3 A vectorized pseudocode to assemble the 3D array $\tilde{\mathbf{B}}$ in Python

Require: `import numpy as np`

`PTW2 = np.matmul($\tilde{\mathbf{P}}$.swapaxes(1, 2), $\tilde{\mathbf{W}} * * 2$)` \triangleright multiplication 3D of $\tilde{\mathbf{P}}^T \tilde{\mathbf{W}}^2$

`$\tilde{\mathbf{A}} = \text{np.matmul}(\text{PTW2}, \mathbf{P})$` \triangleright multiplication 3D of $\tilde{\mathbf{P}}^T \tilde{\mathbf{W}}^2 \tilde{\mathbf{P}}$

`$\tilde{\mathbf{B}} = \text{np.matmul}(\text{inv}(\tilde{\mathbf{A}}), \text{PTW2})$` \triangleright multiplication 3D of $\tilde{\mathbf{A}}^{-1} \tilde{\mathbf{P}}^T \tilde{\mathbf{W}}^2$

The next step is to get the indexes. Considering the right-hand side of (2.15), we can divide the assembly of the global \mathbb{K} matrix into $\mathbb{K} = \mathbb{K}_s + \mathbb{K}_o$, where the matrices \mathbb{K}_s and \mathbb{K}_o are formed by the coefficients of the stars and of the central points, respectively. Therefore, we call \mathbf{B}_{s_g} , \mathbf{I}_{s_g} , and \mathbf{J}_{s_g} the arrays that generate \mathbb{K}_s and \mathbf{B}_{o_g} , \mathbf{I}_{o_g} , and \mathbf{J}_{o_g} those that generate \mathbb{K}_o . The following are the formulas for assembling both matrices:

- Assembly of the \mathbb{K}_s matrix

The vector \mathbf{B}_{s_g} is formed row-wise from $\tilde{\mathbf{B}}$:

$$\mathbf{B}_{s_g} = \left(b_{1,1}^1 \ b_{1,2}^1 \ \dots \ b_{p,m}^1 \ b_{1,1}^2 \ b_{1,2}^2 \ \dots \ b_{p,m}^2 \ \dots \ b_{1,1}^{NI} \ b_{1,2}^{NI} \ \dots \ b_{p,m}^{NI} \right), \quad (3.4)$$

where the super-indices denote the global numbering of the central points of the domain.

$$\mathbf{I}_{s_g} = \left(\underbrace{1 \ \dots \ 1}_{p-m} \ \underbrace{2 \ \dots \ 2}_{p-m} \ \underbrace{3 \ \dots \ 3}_{p-m} \ \dots \ \underbrace{NI \ \dots \ NI}_{p-m} \right) \quad (3.5)$$

and

$$\mathbf{J}\mathbf{s}_g = \left(\overbrace{S_1^1 \dots S_m^1}^{p \text{ times}} \overbrace{S_1^2 \dots S_m^2}^{p \text{ times}} \overbrace{S_1^3 \dots S_m^3}^{p \text{ times}} \dots \overbrace{S_1^{NI} \dots S_m^{NI}}^{p \text{ times}} \right), \quad (3.6)$$

where the set $S_1^k \dots S_m^k$, $k = 1, \dots, NI$, corresponds to the global numbering of the stars. For instance: the specific set $S_1^3 \dots S_{90}^3$ is a star with 90 points whose central point is numbered 3.

Finally,

$$\mathbf{a}\mathbf{s}_g = \left(\overbrace{\underbrace{a_1 a_1 \dots a_1}_m \underbrace{a_2 a_2 \dots a_2}_m \underbrace{a_3 a_3 \dots a_3}_m \dots \underbrace{a_p a_p \dots a_p}_m}_{NI \text{ times}} \right). \quad (3.7)$$

Therefore $\mathbb{K}\mathbf{s}$ is obtained as

$$\mathbb{K}\mathbf{s} = \text{sparse.coo.matrix}((\mathbf{a}\mathbf{s}_g \circ \mathbf{B}\mathbf{s}_g, (\mathbf{I}\mathbf{s}_g, \mathbf{J}\mathbf{s}_g))).$$

- Assembly of the $\mathbb{K}\mathbf{o}$ matrix

To get $\mathbf{B}\mathbf{o}_g$ from $\tilde{\mathbf{B}}$ we use the command `sum($\tilde{\mathbf{B}}$, axis=2, keepdims=True)` in Python. Then, we get a set of 3D arrays

$$\mathbf{B}\mathbf{o}_g = \left(\left(\left(- \sum_{i=1}^m b_{1,i}^0 \dots - \sum_{i=1}^m b_{p,i}^0 \right)^T \left(- \sum_{i=1}^m b_{1,i}^1 \dots - \sum_{i=1}^m b_{p,i}^1 \right)^T \dots \right. \right. \\ \left. \left. \left(- \sum_{i=1}^m b_{1,i}^{NI} - \sum_{i=1}^m b_{2,i}^{NI} \dots - \sum_{i=1}^m b_{p,i}^{NI} \right)^T \right), \quad (3.8)$$

with indexes

$$\mathbf{I}\mathbf{o}_g = \mathbf{J}\mathbf{o}_g = \begin{pmatrix} 0 & 1 & \dots & NI \end{pmatrix}. \quad (3.9)$$

Finally,

$$\mathbf{a}\mathbf{o}_g = \left(\overbrace{(a_1 \ a_2 \ \dots \ a_p)}^{NI \text{ times}} \right). \quad (3.10)$$

Therefore, $\mathbb{K}o$ is obtained as

$$\mathbb{K}o = \text{sparse.coo.matrix}((\text{matmul}(\mathbf{a}o_g, \mathbf{B}o_g), (\mathbf{I}o_g, \mathbf{J}o_g))) + \text{diag}(a_0, a_0, \dots, a_0).$$

Note that $\mathbb{K}o$ is responsible for filling the diagonal of the \mathbb{K} matrix.

To apply the vectorized procedure explained in this section, it is necessary that the number of points per star, m , be the same for all of them.

3.3 Examples

We used 30 and 90 points per star with the distance criterion in the 2D and 3D examples, respectively. In all examples, we used the weighting function $w_i = \|\mathbf{x}_i - \mathbf{x}_0\|_2^{-4}$, $i = 1, 2, \dots, m$.

The Table 3.1 shows the summary of the algorithm. In section 3.3.1 we apply the parallel process in steps 1 and 2. In section 3.3.2 we show the times using the vectorized algorithm in steps 2 and 3.

Table 3.1: Summary of the algorithm developed in GFDM.

Order	Steps of the algorithm
1	star formation
2	derivatives
3	assemble of \mathbb{K}
4	application of boundary conditions
5	solution of the system of equations

3.3.1 Speedup with parallel process

The term speedup refers to the ratio between the processing time with a single core and with multiple cores (six cores in our case). In multiple cores we used a multiprocessing package.

In 2D we solved a square domain with unit edges with 21,609 points obtaining results about three times faster in the specific parts of the GFDM, as shown in the data corresponding to the star formation and derivatives calculation in Table 3.2.

In 3D we solved a cubic domain with unit edges with a total of 26,328 points. The results obtained are more than two times faster in the specific parts of the GFDM, as shown in the data corresponding to the star formation and derivatives calculation in Table 3.3.

Table 3.2: Speedup of each calculation process in the 2D case.

Calculation	one process	six processes	speedup
Star formation	14.35 s	5.17 s	2.77
Derivatives	165.18 s	49.77 s	3.32

Table 3.3: Speedup of each calculation process in the 3D case.

Calculation	one process	six processes	speedup
Star formation	59.44 s	21.28 s	2.79
Derivatives	318.87 s	143.02 s	2.23

3.3.2 Times with vectorized algorithm

Consider a 2D example in a square domain with unit edges and the following partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + 2\frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial x \partial y} + \frac{\partial U}{\partial x} - \frac{\partial U}{\partial y} = 0, \quad (3.11)$$

with exact solution $U = e^x \sin(y)$.

Consider a 3D example in a cubic domain with unit edges and the following partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + 2\frac{\partial^2 U}{\partial z^2} = 0, \quad (3.12)$$

with exact solution $U = e^z \sin(x) \cos(y)$.

Table 3.4: The runtime of the 2D code to calculate the derivatives and assemble the \mathbb{K} matrix.

Algorithm	Total number of points in 2D					
	21,609	29,584	40,804	49,284	63,504	71,289
Non-vectorized	165.18 s	314.01 s	583.84 s	1235.29 s	1407.82 s	1829.25 s
Vectorized	1.10 s	1.70 s	2.43 s	2.75 s	4.02 s	4.84 s

Table 3.5: The runtime of the 3D code to calculate the derivatives and assemble the \mathbb{K} matrix.

Algorithm	Total number of points in 3D			
	8,805	11,663	17,000	26,328
Non-vectorized	43.13 s	68.71 s	138.70 s	318.87 s
Vectorized	2.86 s	4.24 s	6.44 s	19.89 s

Tables 3.4 and 3.5 shows the runtime of the codes to assemble the \mathbb{K} matrix with and without the vectorized algorithm. The results show a considerable saving in execution time with the vectorized algorithm in both applications. As the number of points increases, the computational advantage tends to be more remarkable. In 2D applications, the vectorized algorithm was approximately 150 to 450 times faster than the non-vectorized algorithm. In 3D applications, the vectorized algorithm was approximately 15 to 20 times faster than the non-vectorized algorithm.

Taking into account the calculation of the derivatives and assembling the \mathbb{K} matrix, the steps of execution times using 21,609 points in 2D and 26,328 in 3D are approximately 1s (Table 3.4) and 20s (Table 3.5) in the vectorized algorithm, respectively. In contrast, the parallel process takes approximately 50s and 140s only to calculate the derivatives (see Tables 3.2 and 3.3). Therefore, combining the parallel process in the star formation with the vectorized algorithm in the calculation of the derivatives is the most suitable way.

We highlight the following additional comments:

- In the 3D example, the non-vectorized algorithm failed to assemble the \mathbb{K} matrix for approximately 30,000 points. The failure occurred due to insufficient memory when executing the `sparse.coo.matrix` command. In the vectorized algorithm, the same command is executed but requires less memory because the assembly of \mathbb{K} is split into two steps: one for the \mathbb{K}_s matrix and another for the \mathbb{K}_o matrix.

Table 3.6: Times to the non-vectorized and vectorized codes in 2D and 3D examples.

Stages	2D example (70,000 points)		3D example (25,000 points)	
	Non-vectorized	Vectorized	Non-vectorized	Vectorized
Star formation	181 s	181 s	38 s	38 s
Derivatives and assembly of \mathbb{K}	884 s	4 s	315 s	20 s
Solve	35 s	35 s	108 s	108 s

- Table 3.6 shows the times in 2D and 3D examples at different stages of the codes using approximately 70,000 and 25,000 points, respectively. In the non-vectorized algorithm, the assembly of the \mathbb{K} matrix is clearly the bottleneck of the code. With the vectorized algorithm, this bottleneck is eliminated. Then, in 2D and 3D, the formation of the stars and the solution of the system of equations become the most time-consuming stages ¹, respectively.

- Note that by changing the partial differential equation, only the elements of (2.18) are modified. However, the size of this array only depends on the order of approximation and the dimension (1D, 2D, or 3D) chosen. Then, the times calculated here should be approximately equal by varying only the partial differential equation.

¹Note that in 3D the number of non-zero elements in each row of \mathbb{K} is much greater than in 2D, affecting considerably the time in solving the system of equations.

Chapter 4

Treatment of ill-conditioned stars

4.1 Strategy to avoid ill-conditioned stars

A drawback of the GFDM is the possibility of generating ill-conditioned stars through the inverse matrices in (2.12). The stability of the solution depends on the condition number of the matrix involved in the approximation of the derivatives. A large condition number warns us that special care may be necessary when solving the system of linear equations.

Since the parameters of the method can be chosen arbitrarily, an appropriate change in these parameters can significantly reduce the condition number. The key issue is to detect troublesome stars in advance.

First, we need to establish a tolerance for each star from which to consider that the star has a large condition number. After that, we only act on those stars that exceed this tolerance value.

Given a discretization, we form all the stars with the desired number of points m_0 using the distance criterion. When applying the algorithm not all stars will end up with the same number of points, from here on, we denote with m_0 the initial number of points of each star. For each of those stars, we calculate the tolerance as follows:

1. Calculate the minimum distance d_i between any point on the star and the central point, obtaining a vector \mathbf{d} of minimum distances (see step 1 at Figure 4-1).
2. Consider λ intervals $[\delta_i, \delta_{i+1})$ with $i = 1, 2, \dots, \lambda$ such that the interval $[d_{min}, d_{max}]$

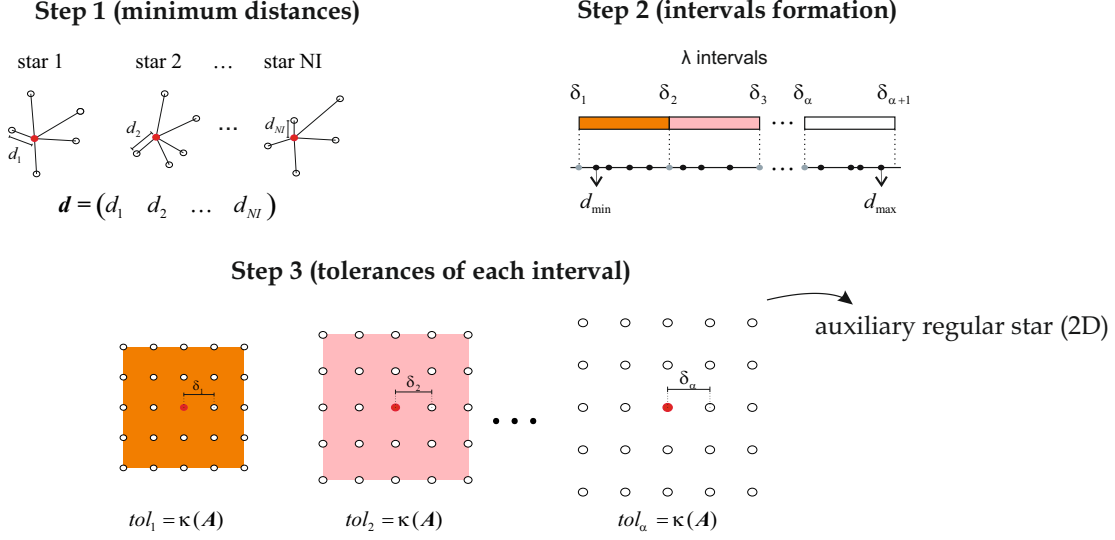


Figure 4-1: Strategy to avoid large condition number in the matrix \mathbf{A} .

$\subset \cup_{i=1}^{\lambda} [\delta_i, \delta_{i+1})$ (see step 2 at Figure 4-1, where the black points represent the minimum distances obtained in step 1 and the gray points represent the minimum distance of the auxiliary regular stars). In particular, the criterion that we apply is to divide intervals of consecutive powers of 10, $[10^j, 10^{j+1})$, $j \in \mathbb{Z}$, into 10 equal parts.

3. Once obtained the minimum values δ_i of the intervals, calculate the condition numbers $\kappa(\mathbf{A})$ of the auxiliary regular stars with stepsize δ_i in all directions (see step 3 at Figure 4-1, where the tolerance of the auxiliary regular star with stepsize δ_i represents all the stars whose minimum distances are in the interval $[\delta_i, \delta_{i+1})$). The condition numbers obtained are the respective tolerances tol_i with $i = 1, 2, \dots, \lambda$. We say that a star is regular when their points are located on a square (2D) or on a cube (3D) contained in a regular grid. In particular, we use stars with 24 points in 2D and 124 points in 3D because they are the first square (2D) and the first cube (3D) whose number of points is greater than the minimum required for the approximation order.

Note that choosing the minimum value in step 3 for each interval is a conservative choice as the tolerance is the highest one in that case.

We consider an example to clarify these steps. Given a discretization with 5 stars, we calculate the minimum distances and they are 0.091, 0.089, 0.083, 0.175 and 0.151 (step 1). Then we consider the 3 intervals $[0.08, 0.09)$, $[0.09, 0.1)$ and $[0.1, 0.2)$ whose

union contains $[d_{min}, d_{max}] = [0.083, 0.175]$ (step 2). Finally, we calculate the condition numbers (the tolerances) for the 3 regular stars with stepsizes 0.08 (tol_1), 0.09 (tol_2) and 0.1 (tol_3) in all directions and so, the stars with minimum distance equal to 0.089 and 0.083 have tolerance tol_1 , the star with minimum distance equal to 0.091 has tolerance tol_2 and the stars with minimum distance equal to 0.175 and 0.151 have tolerance tol_3 (step 3).

After calculating the tolerances, each star has its corresponding associated tolerance and if the condition number of \mathbf{A} is greater than such tolerance, then the star can be corrected as follows:

- Stage 1: In both 2D and 3D cases, the distance criterion is changed by the quadrant criterion (2D) or by the octant criterion (3D).
- Stage 2: If the condition number of the recalculated matrix \mathbf{A} is still greater than the corresponding tolerance tol_i , then we add m_0 points to the star in 2D, or $2m_0$ points to the star in 3D.

It is important to note that this strategy uses known techniques to correct the stars that are at risk of being ill-conditioned. The key issue is that this strategy allows deciding which stars are formed in one way or another, contrary to the usual practice of using a fixed number of points per star and the same formation criterion for all stars in the domain.

4.2 Numerical results

This section is devoted to the numerical experiments to assess the validity of the proposed strategy. We show the performance of the strategy to detect and treat ill-conditioned stars for 2D and 3D in sections (4.2.1) and (4.2.2), respectively. We study in section (4.2.3) the influence on the proposed strategy of different weighting functions with regard to the accuracy and the number of modified stars. Finally, in section (4.2.4) we study the influence on the proposed strategy of the number of intervals.

The 2D (D1, D2, D3 and ID1) and 3D (D4, D5, D6, and ID4) domains used in the current section are in Figures 2-2 and 2-4, respectively.

In all cases, and for each star, we use the weighting function $w_i = \|\mathbf{x}_i - \mathbf{x}_0\|_2^{-4}$, $i = 1, 2, \dots, m$. The distance criterion is used in all cases, except for the stars where we applied the correction.

We use the following global error formula for the errors:

$$\text{Global error (\%)} = \sqrt{\frac{\sum_{j=1}^{\text{NI}} (U_j - u_j)^2}{\text{NI}}} \cdot 100, \quad (4.1)$$

where U_j is the exact value of the solution at the point $(x_j, y_j, z_j) \in M \cap \Omega$ and u_j is the approximate solution provided by the GFDM at the same point¹.

4.2.1 Results with the proposed strategy to avoid ill-conditioned stars in 2D

We apply the proposed strategy using the minimum number of points per star or close to it to show the efficiency of the algorithm when dealing with ill-conditioned stars.

To calculate the tolerances from which to apply the algorithm we have needed 1, 4, 5 and 5 auxiliary regular stars in D1, D2, D3 and ID1, respectively.

As an example, in D2 we have used the intervals $[0.04, 0.05)$, $[0.05, 0.06)$, $[0.06, 0.07)$ and $[0.07, 0.08]$ to generate regular stars with minimum distances of 0.04, 0.05, 0.06 and 0.07. The condition number for each of these stars is the tolerance shared by all stars that have their minimum distance in the associated interval.

For the third-order approximation, we show in Table 4.1 the global error obtained when using stars with 20 points, which we use as a reference value, stars with 9 points formed by the distance criterion and stars with 9 points applying the proposed strategy to avoid ill-conditioned stars.

¹For convenience, the numbering used here is global, as distinct from the local numbering u_i , $i = 0, 1, \dots, m$, used for each star in chapter 2.

Table 4.1: Global errors with the third-order approximation in 2D

Equation	Discretization	third-order approximation		
		$m = 20$	$m = 9$	$m_0 = 9$ (with strategy)
(2.22)	D1	1.20e-2	357.14	1.22e-2
(2.23)	D2	3.12e-2	2.82e-1	3.09e-2
(2.24)	D3	7.05e-2	39.50	9.26e-2
(2.22)	ID1	4.00e-2	73.50	2.98e-2

Table 4.2: Global errors with the fourth-order approximation in 2D

Equation	Discretization	fourth-order approximation		
		$m = 30$	$m = 14$	$m_0 = 14$ (with strategy)
(2.22)	D1	1.10e-4	72.70	5.45e-4
(2.23)	D2	1.71e-2	427.05	3.65e-2
(2.24)	D3	2.68e-3	1635.12	5.96e-3
(2.22)	ID1	4.12e-3	260.90	2.34e-2

For the fourth-order approximation, we show in Table 4.2 the global error obtained when using stars with 30 points, which we use as a reference value, stars with 14 points formed by the distance criterion and stars with 14 points applying the proposed strategy to avoid ill-conditioned stars.

Both cases show that the errors applying the proposed strategy are of the same order or less than the errors obtained using the number of points established as a reference, which is a greater number, as a consequence of the treatment given to the ill-conditioned stars.

Note that the lower the number of initial points m_0 , the greater the possibility of finding ill-conditioned stars and, therefore, a greater number of stars must be treated. For example, for the equation (2.22) with the discretization used on D1 and fourth-order approximation, the second stage of the algorithm was reached in 96% of the points.

However, if we increase the number of initial points, we reduce the possibility of the appearance of ill-conditioned stars and therefore fewer stars should be treated, which is a desirable situation. In the previous example (equation (2.22) with the discretized domain D1 in the fourth-order approximation), if we increase the number of initial

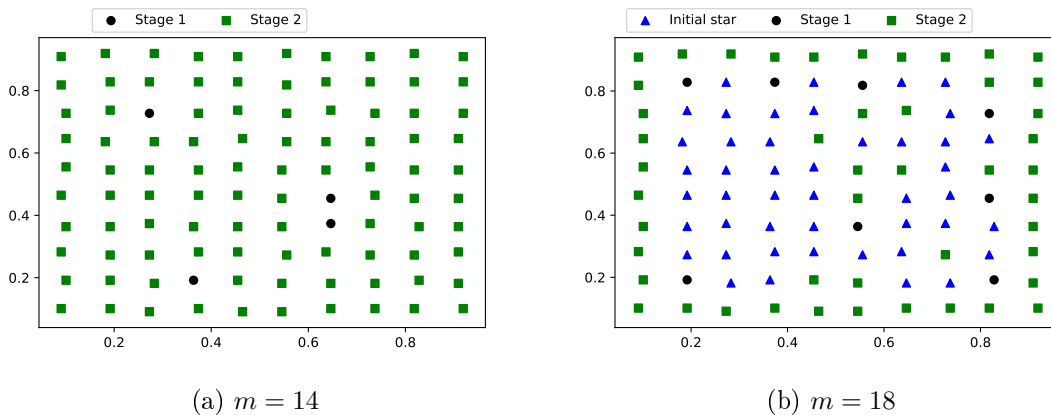


Figure 4-2: Distribution of the formation of stars used at each point in the discretization: stage 1 (quadrant criterion) and stage 2 (quadrant criterion with the addition of points).

points from 14 to 18, still far from the reference 30, the number of stars that reach the second stage is 47%, 8% of the stars remains in the first stage and the remaining 45% does not need treatment (those stars were formed by the distance criterion). Furthermore, the global error obtained in this case without applying the strategy is 85.34% and applying the strategy is 2.27e-4%, less than the error of 5.45e-4% shown in Table 4.2.

Figure 4-2 shows the stages reached by the points of the discretization $D1$ when we consider 14 initial points (a) and 18 initial points (b). We want to highlight that all the interior points close to the boundary reach the second stage of the algorithm. One way to reduce the appearance of ill-conditioned stars in this case would be to make a more adequate distribution of the boundary points, but we have not considered this issue in this thesis.

We are proposing a number of points as a reference for a type of irregular discretization that has a defined pattern and is applicable to any domain. In any case, we have also used a completely irregular discretization with randomly distributed points. For the latter case, we show the errors caused both by applying the strategy and without applying it to the highest errors, as can be seen in Tables 4.3 and 4.4 for the third- and fourth-order approximations, respectively. Note that these errors are in line with the errors of the discretization $D1$.

Table 4.3: Global error with third-order approximation in 2D for the discretization ID1.

m_0	Without strategy	With strategy
9	73.50	2.98e-2
10	28.45	2.98e-2
11	27.65	1.43e-1

Table 4.4: Global error with fourth-order approximation in 2D for the discretization ID1.

m_0	Without strategy	With strategy
14	260.90	2.34e-2
15	18.78	2.03e-3
16	75.66	1.97e-3
17	37.92	1.32e-3
18	8.55e-1	1.64e-3

4.2.2 Results with the proposed strategy to avoid ill-conditioned stars in 3D

As we did in the 2D cases, we apply the proposed strategy using the minimum number of points per star or close to it to show the efficiency of the algorithm for treating ill-conditioned stars.

To calculate the tolerances from which to apply the algorithm we have needed 1 auxiliary regular star both in D4 and D5, 6 auxiliary regular stars in D6, and 8 auxiliary regular stars in ID4.

For the third-order approximation, we show in Table 4.5 the global error obtained when using stars with 50 points, which we use as a reference value, stars with 19 points formed by the distance criterion and stars with 19 points applying the proposed strategy to avoid ill-conditioned stars.

For the fourth-order approximation, we show in Table 4.6 the global error obtained when using stars with 90 points, which we use as a reference value, stars with 36 points formed by the distance criterion and stars with 36 points applying the proposed strategy to avoid ill-conditioned stars.

Table 4.5: Global error with the third-order approximation in 3D

Equation	Discretization	third-order approximation		
		$m = 50$	$m = 19$	$m_0 = 19$ (with strategy)
(2.25)	D4	1.07e-2	1.55e-1	9.71e-3
(2.26)	D5	1.94e-2	127.58	1.11e-2
(2.27)	D6	3.43e-2	4.05	4.60e-2
(2.25)	ID4	8.65e-2	10.60	3.05e-2

Both cases show that the errors applying the proposed strategy are of the same order or less than the errors obtained using the number of points established as a reference, which is a greater number, as a consequence of the treatment given to the ill-conditioned stars.

Table 4.6: Global error with the fourth-order approximation in 3D

Equation	Discretization	fourth-order approximation		
		$m = 90$	$m = 36$	$m_0 = 36$ (with strategy)
(2.25)	D4	2.30e-3	0.101	3.76e-3
(2.26)	D5	3.79e-3	154.94	4.45e-3
(2.27)	D6	1.05e-2	63.28	3.18e-2
(2.25)	ID4	5.76e-3	2.04e-2	1.26e-2

As in 2D cases, the lower the number of initial points, the greater the possibility of finding ill-conditioned stars and, therefore, a greater number of stars must be treated. For example, for the equation (2.26) in the discretization of the domain D5 with fourth-order approximation, the second stage of the algorithm was reached in 82% of the points.

However, if we increase the number of initial points, we reduce the possibility of the appearance of ill-conditioned stars and, therefore, fewer stars should be treated, which is a desirable situation. In the previous example (equation (2.26) in the discretized domain D5), if we increase the number of initial points from 36 to 42, still far from the reference 90, the number of stars that reach the second stage is 39%, 9% of the stars remains in the first stage and the remaining 52% do not need treatment (these stars are formed by the distance criterion). Moreover, the global error obtained in this

Table 4.7: Global error with the third-order approximation in 3D for the discretization ID4.

m_0	Without strategy	With strategy
19	10.60	2.82e-2
64	1.35	2.48e-2
65	2.74	3.05e-2

Table 4.8: Global error with the fourth-order approximation in 3D for the discretization ID4.

m_0	Without strategy	With strategy
41	1.14e-1	2.05e-2
44	2.49e-1	5.27e-3
50	1.02e-1	5.05e-3

case without applying the strategy is 52.49% and applying the strategy is 2.67e-3%, less than the error of 4.45e-3% shown in Table 4.6.

We are proposing a number of points as a reference for a type of irregular discretization that has a defined pattern and is applicable to any domain. In any case, we have also used a completely irregular discretization with randomly distributed points. For the latter case, we show the errors caused both by applying the strategy and without applying it to the highest errors, as can be seen in Tables 4.7 and 4.8 for the third- and fourth-order approximations, respectively. Note that these errors are in line with the errors of the discretization D4.

4.2.3 Influence of the weighting function

The weighting function affects both the accuracy of the solution and the number of stars that must be modified by the proposed algorithm for a fixed number of auxiliary stars.

To analyze this, we consider potential and exponential weighting functions that have already given good results before [8] and solve the equation (2.22) in the domains D1 and ID1 with $m_0 = 18$ and the equation (2.25) in the domains D4 and ID4 with $m_0 = 42$ using in both cases the fourth-order approximation.

In Table 4.9 we show the weighting functions used, the global error and the percentage of stars that have been modified by the algorithm.

We note that, in general, the potential weighting functions provide better accuracy, but the number of modified stars is greater as the exponent increases. On the other hand and also in a general way, the exponential weighting functions need to modify a smaller number of stars but have a lower accuracy. A good balance can be given for potential weighting functions with exponent 2 or 4, although this will depend on the needs of the problem to be solved.

Table 4.9: Relation between the weighting functions, the global error and the percentage (%) of modified stars.

	Domain D1		Domain ID1		Domain D4		Domain ID4	
weighting function	error (%)	% of modified stars	error (%)	% of modified stars	error (%)	% of modified stars	error (%)	% of modified stars
$\ \mathbf{x}_i - \mathbf{x}_0\ _2^{-2}$	1.90e-4	51.00	1.03e-2	59.62	5.89e-3	20.00	3.31e-2	2.97
$\ \mathbf{x}_i - \mathbf{x}_0\ _2^{-4}$	2.30e-4	55.00	1.65e-3	87.50	6.45e-3	36.00	5.09e-2	61.49
$\ \mathbf{x}_i - \mathbf{x}_0\ _2^{-6}$	6.20e-4	98.00	6.30e-4	100.00	1.81e-3	38.40	4.82e-1	89.63
$\ \mathbf{x}_i - \mathbf{x}_0\ _2^{-8}$	2.58e-3	100.00	1.18e-3	100.00	1.27e-2	40.80	3.01e0	91.86
$e^{-4\ \mathbf{x}_i - \mathbf{x}_0\ _2^2}$	5.53e-3	41.00	8.12e-3	40.39	1.09e-2	0.80	2.34e-2	0.00
$e^{-9\ \mathbf{x}_i - \mathbf{x}_0\ _2^2}$	1.80e-3	41.00	3.44e-2	40.39	1.94e-2	0.80	4.90e-3	0.75
$e^{-16\ \mathbf{x}_i - \mathbf{x}_0\ _2^2}$	5.60e-4	44.00	1.95e-2	40.39	2.35e-1	4.80	1.35e-2	1.49

4.2.4 Influence of the number of intervals on the proposed strategy

The number of intervals, λ , affects the number of stars modified by the algorithm, so it also affects the accuracy. The number of intervals coincides with the number of auxiliary stars to be calculated additionally and this also affects the execution time, but the number of extra stars is so small that we can neglect it.

To analyze this, we solve the equation (2.22) in the domains D1 and ID1 with $m_0 = 18$ and the equation (2.25) in the domains D4 and ID4 with $m_0 = 42$ using

in both cases the fourth-order approximation. We consider the value of λ obtained according to step 2 of section 4.1 and three refinements: 2λ , 4λ , and 8λ .

Given the results shown in Table 4.10, we can say that increasing the number of intervals does not have a significant impact on increasing the number of modified stars or reducing the error. The only exception would be when the discretization is practically regular because in that case the condition numbers of the stars will be very close to the tolerances and a small increase in the number of intervals may cause a large increase in the number of modified stars, as occurs in D4. Therefore, the strategy to λ proposed in this thesis is sufficient.

Table 4.10: Relation between the number of intervals λ , the global error and the percentage (%) of modified stars.

Domain D1			Domain ID1		
Number of intervals	error (%)	% of modified stars	Number of intervals	error (%)	% of modified stars
$\lambda_0 = 2$	2.27e-4	54.00	$\lambda_0 = 6$	1.64e-3	87.50
$2\lambda_0 = 4$	2.26e-4	54.00	$2\lambda_0 = 12$	1.39e-3	96.15
$4\lambda_0 = 8$	2.26e-4	54.00	$4\lambda_0 = 24$	1.53e-3	98.08
$8\lambda_0 = 16$	2.16e-4	56.00	$8\lambda_0 = 48$	1.48e-3	99.38
Domain D4			Domain ID4		
Number of intervals	error (%)	% of modified stars	Number of intervals	error (%)	% of modified stars
$\lambda_0 = 1$	6.45e-3	36.00	$\lambda_0 = 9$	5.09e-2	61.48
$2\lambda_0 = 2$	4.19e-3	78.40	$2\lambda_0 = 18$	3.92e-2	68.15
$4\lambda_0 = 4$	3.36e-3	79.20	$4\lambda_0 = 36$	1.61e-2	72.59
$8\lambda_0 = 8$	3.37e-3	80.00	$8\lambda_0 = 72$	9.65e-3	75.55

Chapter 5

A technique for generating adapted discretizations

5.1 A procedure for generating discretizations adapted to the partial differential equation

Our goal is to obtain a discretization adapted to the problem to be solved. The strategy consists of solving the problem twice but with different resolution purposes and different discretizations. In the first step, we use a coarse uniform discretization and compute the absolute values of the gradients. Depending on these values, we place more points where the gradients are higher and fewer points where the gradients are lower. In the second stage, we calculate the approximate solution using the adapted discretization generated in the previous stage. To generate the adapted discretization and for each point of the initial discretization, we consider an area of influence where the new points are placed.

It is important to highlight several aspects:

- The use of a regular discretization as a starting point is due to the fact that better results are obtained than using random discretizations [8, 65].
- The initial discretization is, in general, locally regular and globally irregular, since we only discretize regularly in the interior of the domain.

- The resulting adapted discretization is also locally irregular, which do not impose any limitation to discretize any problem.

We have divided this section into two parts, addressing the generation of interior points and the generation of boundary points. In the first part, we distinguish between interior points generated from interior points and from boundary points.

5.1.1 Procedure for generating interior points

Firstly, we use a coarse uniform discretization to get an initial approximate solution by applying a second-order approximation with the GFDM. This initial coarse discretization, should be fine enough to provide a correct solution, although not necessarily with great accuracy. With these results, we compute the euclidean norm of the vector gradient at each interior central point, that is

$$\|\vec{\nabla}_0\| = \sqrt{\left(\frac{\partial u_0}{\partial x}\right)^2 + \left(\frac{\partial u_0}{\partial y}\right)^2}. \quad (5.1)$$

Each interior point of the domain has the same square area of influence, with the geometric center at the respective point (see Figure 5-1). The length of the side of this square, Δ , is the shortest distance between two points in all the domain.

Let $\bar{\nabla}$ be the average gradient, σ the standard deviation of the gradients and β a constant. The general rule for deciding the number of points ¹ in each area of influence is to place regular square grids with 1, 4 or 9 points (see Figure 5-2) depending on the gradient values as follows:

- i) 1 point if $\|\vec{\nabla}_0\| < \bar{\nabla} + \beta\sigma$;
- ii) 4 points if $\bar{\nabla} + \beta\sigma \leq \|\vec{\nabla}_0\| < \bar{\nabla} + 2\beta\sigma$;
- ii) 9 points if $\|\vec{\nabla}_0\| \geq \bar{\nabla} + 2\beta\sigma$.

If the influence area has 4 points, then we have a distance between them of $h = \Delta/3$ and if it has 9 points, the distance is $h = \Delta/4$. To facilitate a smooth transition

¹We considered the possibility of using four regular square grids, the previous ones and also the one with 16 points. However, we observed that this additional grid of 16 points did not improve the accuracy of the results significantly. Therefore, we have decided to implement the method with these three types of grids and thus avoid a large number of points in the adapted discretization.

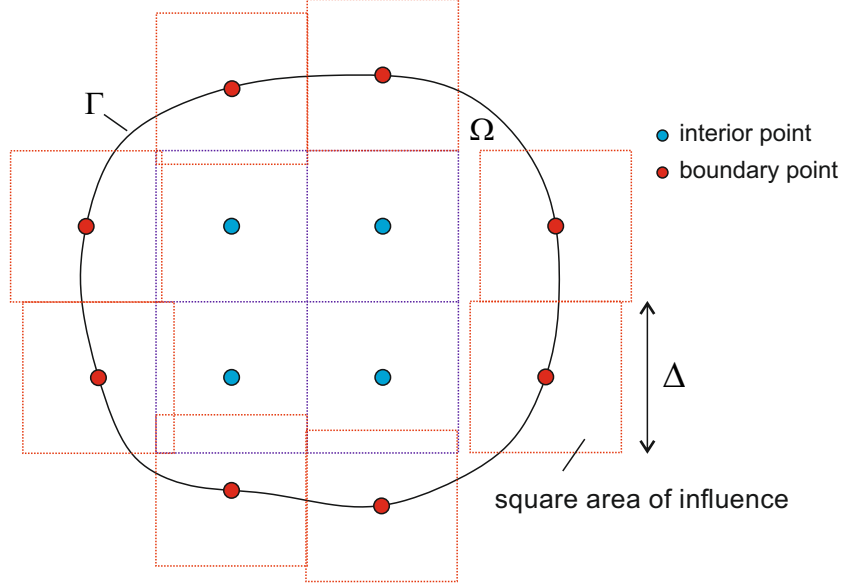


Figure 5-1: Influence areas for each point in the domain with their geometric centres at the respective points. Note that there is no overlap between the influence areas relative to the internal points. However, overlap can occur between the influence area of a interior point and the influence area of a boundary point.

between the different influence areas, we multiply h by an expansion coefficient α , checking that the points do not leave their influence area. This α is given by

$$\alpha = 1 + \frac{2\eta}{n-1}, \quad (5.2)$$

where η is an expansion percentage ($0 \leq \eta \leq 1$) and n is the number of points within the area of influence in one direction ($n = 2$ in case of 4 points and $n = 3$ in case of 9 points). So, α is just a percentage of broadening of the regular grid in the influence area. We give below the details of how α is obtained.

Since the Δ value is invariable considering the same discretization M , then we established a limit in α to avoid that any point be inserted outside the area of influence to which it belongs.

Considering $\alpha \geq 1$, we have

$$(n-1)h \leq \sum_{i=1}^{n-1} \alpha h \leq \Delta \Rightarrow (n-1)h \leq \alpha h(n-1) \leq \Delta. \quad (5.3)$$

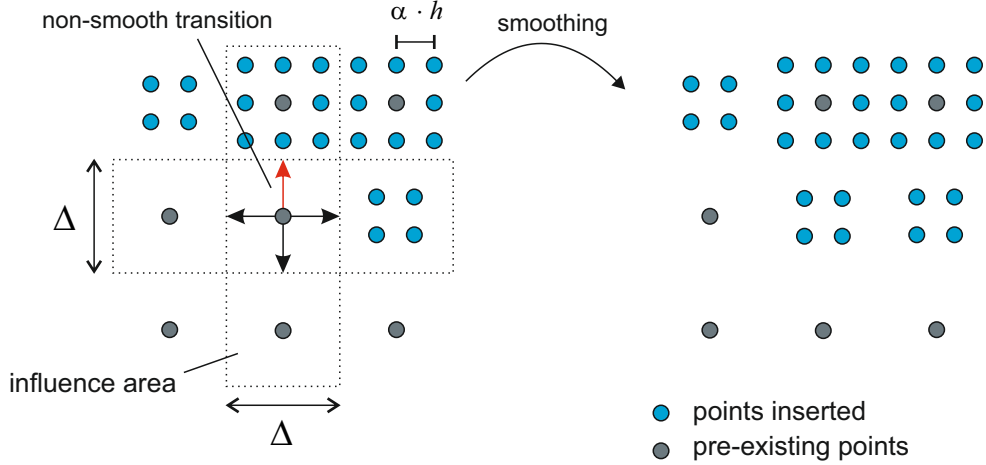


Figure 5-2: The insertion of points within the areas of influence (left) and the discretization smoothing process (right).

Since $h = \frac{\Delta}{n+1}$, then it is

$$(n-1)\frac{\Delta}{n+1} \leq \alpha\Delta\frac{n-1}{n+1} \leq \Delta, \quad (5.4)$$

from which it follows that

$$1 \leq \alpha \leq \frac{n+1}{n-1}. \quad (5.5)$$

Finally, we adopted an expansion coefficient given by

$$\alpha = 1 + \eta(\max(\alpha) - \min(\alpha)) = 1 + \frac{2\eta}{n-1}. \quad (5.6)$$

After inserting the corresponding points in the influence areas, we check if there are influence areas with a single point and at least one of the four closest areas of influence has 9 points. In such a case, we treat the area of influence of the single point as if it were the case with 4 points. In this way, we smooth the transition between influence areas, as can be seen in Figure 5-2.

Further, since we do not have the gradient values for the boundary points, we associate to each boundary point the number of points containing the area of influence of the closest interior point. Then we place the points as before but without considering

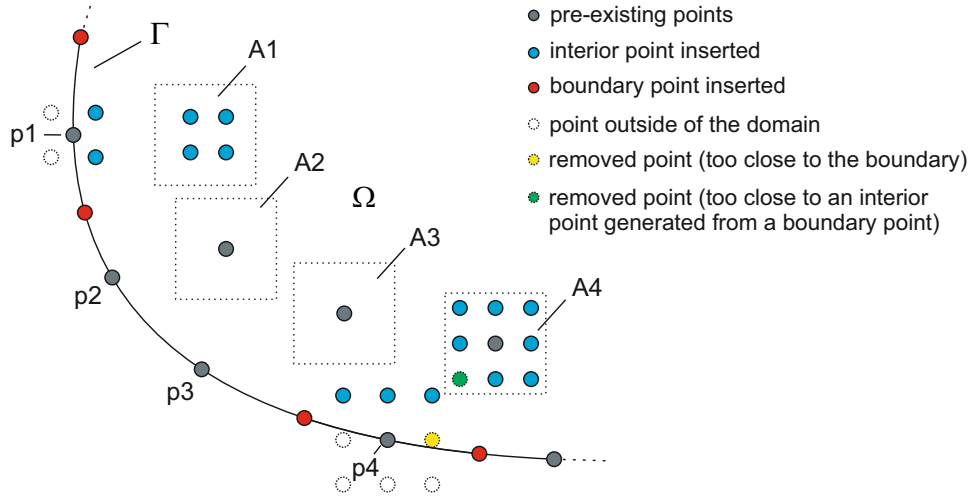


Figure 5-3: Interior points generated from boundary points and boundary points inserted. The points p_1 , p_2 , p_3 , and p_4 have the closest interior areas A_1 , A_2 , A_3 , and A_4 with 4, 1, 1, and 9 points, respectively. The white points are rejected because they are outside the domain, the yellow point is removed because it is too close to the boundary, and the green point is removed because it is close to an interior point generated from a boundary point. The blue and red points are the interior and boundary points inserted, respectively.

the points that fall outside the domain and that are located at a distance less than $\Delta/10$ from the boundary. We also use two additional restrictions in this case, on the one hand, an expansion percentage (η) 50% higher than that used in the previous case and, on the other hand, if one of the interior points added from the boundary is less than $\Delta/2$ from any other interior point, we keep it and remove the other interior point. Note that the inserted points in this section are not in Γ (see Figure 5-3).

5.1.2 Procedure for generating boundary points

We have just explained in the last paragraph of subsection 5.1.1 that the boundary points will have in their area of influence 1, 4, or 9 points which will be determined by the number of points in the area of influence of the closest interior point. In this way, we can add interior points near the boundary. Now we deal with how to add points on the boundary itself.

In order to add points on the boundary, we use a cubic spline interpolation.

If the considered boundary point has been treated in the previous step as an area

of influence of 1 point, then we do not add points, and, if it has been treated as an area of influence of 4 or 9 points, then we add two points, one on each side of the considered boundary point and at half distance with respect to the nearest boundary points. For example, in Figure 5-3, boundary points p1 and p4 generate 4 and 9 points, respectively, in their areas of influence, and consequently, we add two points (red points) in each case. The boundary points p2 and p3 do not generate points in their areas of influence, so nothing is done. Of course, in case of overlapping points in Γ , only one of them is added.

5.2 Numerical results

In all examples, we used the algorithm of star correction (chapter 4). In particular, we used the weighting function $w_i = \|\mathbf{x}_i - \mathbf{x}_0\|_2^{-4}$, $i = 1, 2, \dots, m$, and stars with 18 initial points formed by the distance criterion. In section 5.2.4, we used the second-order approximation and stars with 16 points formed by the distance criterion.

In all examples, we chose $\eta = 0.25$, $\beta = 0.5$ and have considered two uniform discretizations whose errors are above and below the error obtained with the corresponding adapted discretization.

We used the following global error formula:

$$\text{Error}(\%) = \frac{\sqrt{\frac{\sum_{j=1}^{\text{NI}} (U_j - u_j)^2}{\text{NI}}}}{u_{\max}} \cdot 100, \quad (5.7)$$

where U_j is the exact value of the solution at the point $(x_j, y_j) \in M \cap \Omega$, u_j is the approximate solution provided by the GFDM at the same point¹, and u_{\max} is the maximum value of the approximations given by the GFDM.

In all examples, by uniform discretization we mean a uniform discretization over the entire domain except for the boundaries.

¹The numbering used here is global, as distinct from the local numbering u_i , $i = 0, 1, \dots, m$, used for each star in section 2.

We wrote the codes in Python and used an Intel i7-8750H processor to execute the codes. The calculated values of speedups in each example refer to the ratio between the runtime using a uniform discretization and the one of the corresponding adapted discretization. Of course, when we put the runtime of an adapted discretization, the calculation of the gradients was also included.

In section 5.2.1, we show the errors and the code runtimes in adapted discretizations. We study the influence on the proposed technique for different values of η and β concerning the accuracy in section 5.2.2. We study the influence on the proposed technique for different values of weighing functions concerning the accuracy in section 5.2.3. Finally, in section 5.2.4, we show the errors and the code runtimes of adapted discretizations with second-order approximations.

5.2.1 Adapted discretizations with fourth-order approximation

Example 1

Consider the square domain

$$D7 = \{(x, y) \in \mathbb{R}^2 | 0.01 \leq x, y \leq 1.01\}$$

and the boundary value problem given by the following partial differential equation:

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 2500e^{-50x} + 25e^{-5+5x} + 25e^{-5y} + 25e^{-5+5y} \quad (5.8)$$

whose exact solution is

$$U(x, y) = e^{-50x} + e^{-5y} + e^{-5+5x} + e^{-5+5y}. \quad (5.9)$$

We solve (5.8) in the domain D7 considering an initial discretization with 196 points. The gradient field obtained from the initial discretization is shown in Figure 5-4a. We denote by SA1 the resulting adapted discretization that can be seen in

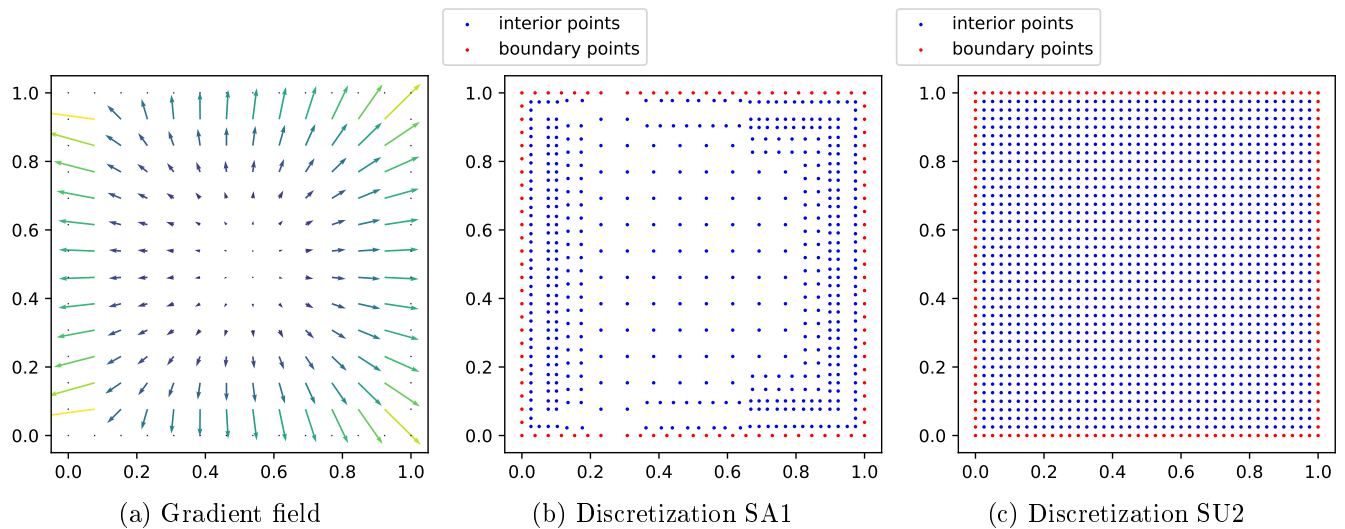


Figure 5-4: Gradients, adapted and uniform discretizations for Example 1.

Table 5.1: Global error and execution times for Example 1

Discretization	Total number of points (N)	Error (%)	Time (s)
SU1	1600	8.63e-1	7.02
SU2	1681	7.93e-1	7.92
SA1 (adapted)	596	8.10e-1	2.05

Figure 5-4b with 596 points. The uniform discretizations SU1 and SU2 have 1600 and 1681 points, respectively. The uniform discretization SU2 is shown in Figure 5-4c.

Table 5.1 shows the errors and the execution times of all discretizations. In the adapted discretization SA1, we achieved similar accuracy with a decrease of approximately 65% in the number of points and with a speedup of approximately 4 relative to the uniform discretization SU2.

Example 2

Consider the boundary value problem given by the following partial differential equation:

$$\frac{\partial^2 U}{\partial x^2} - \frac{\partial^2 U}{\partial y^2} = 0, \quad (5.10)$$

whose exact solution is

$$U(x, y) = e^{-x-y-2}. \quad (5.11)$$

We solve (5.10) in the domain D7 using an initial discretization with 220 points. The gradient field obtained from the initial discretization is shown in Figure 5-5a. We denote by SA2 the adapted discretization that can be seen in Figure 5-5b with 526 points. The uniform discretizations SU3 and SU4 have 2254 and 2304 points, respectively. The discretization SU4 is shown in Figure 5-5c.

Table 5.2 shows the errors and the execution times of all discretizations. In the adapted discretization SA2, we achieved similar accuracy with a decrease of approximately 75% in the number of points and with a speedup of approximately 6 relative to the uniform discretization SU4.

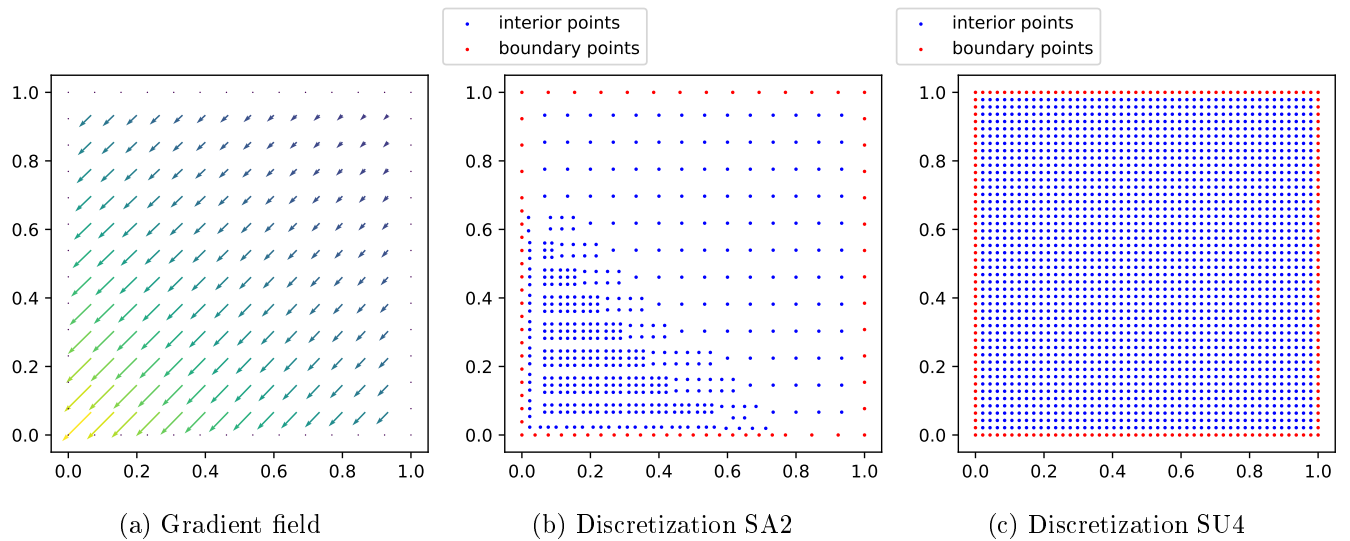


Figure 5-5: Gradients, adapted and uniform discretizations for Example 2.

Table 5.2: Global error and execution times for Example 2

Discretization	Total number of points (N)	Error (%)	Time (s)
SU3	2254	5.96e-1	11.00
SU4	2304	1.94e-2	11.19
SA2 (adapted)	526	3.34e-2	1.79

Example 3

Consider the circular domain

$$D8 = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1.5^2\}$$

and the boundary value problem given by the following partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \frac{4(x^2 + y^2 - 0.01)}{(x^2 + y^2 + 0.01)^3}, \quad (5.12)$$

with exact solution

$$U(x, y) = \frac{1}{x^2 + y^2 + 0.01}. \quad (5.13)$$

We solve (5.12) in the domain D8 using an initial discretization with 488 points. The gradient field obtained from the initial discretization is shown in Figure 5-6a. We denote by CA1 the adapted discretization that can be seen in Figure 5-6b with 696 points. The uniform discretizations CU1 and CU2 have 1719 and 1799 points, respectively. The discretization CU2 is shown in Figure 5-6c.

Table 5.3 shows the errors and the execution times of all discretizations. In the adapted discretization CA1, we achieved similar accuracy with a decrease of approximately 60% in the number of points and with a speedup of approximately 4 relative to the uniform discretization CU1.

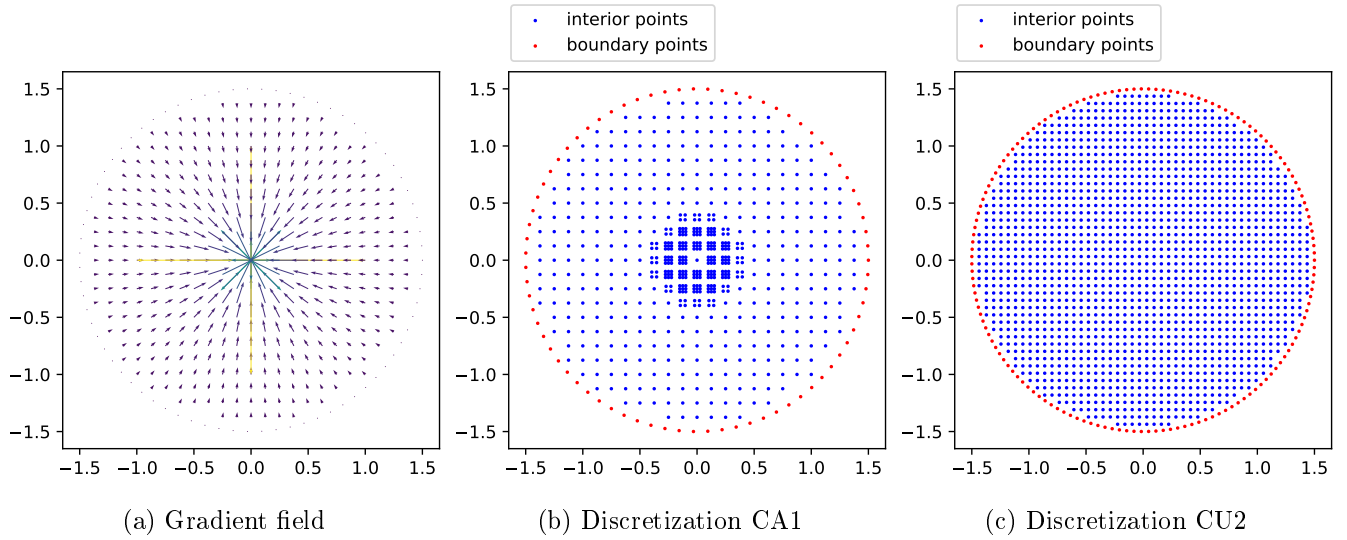


Figure 5-6: Gradients, adapted and uniform discretizations for Example 3.

Table 5.3: Global error and execution times for Example 3

Discretization	Total number of points (N)	Error (%)	Time (s)
CU1	1719	9.61e-1	15.94
CU2	1799	6.17e-1	21.75
CA1 (adapted)	696	9.38e-1	4.29

Example 4

Consider the boundary value problem given by the following partial differential equation

$$\begin{aligned}
 \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = & \frac{8x^2}{(x^2 + y^2 + 0.01)^3} - \frac{4}{(x^2 + y^2 + 0.01)^2} + \frac{2(2x + 1.50)^2}{((x + 0.75)^2 + (y + 0.75)^2 + 0.01)^3} \\
 & - \frac{4}{((x + 0.75)^2 + (y + 0.75)^2 + 0.01)^2} + \frac{8y^2}{(x^2 + y^2 + 0.01)^3} + \frac{2(2y + 1.50)^2}{((x + 0.75)^2 + (y + 0.75)^2 + 0.01)^3}
 \end{aligned}
 \tag{5.14}$$

with exact solution

$$U(x, y) = \frac{1}{x^2 + y^2 + 0.01} + \frac{1}{(x + 0.75)^2 + (y + 0.75)^2 + 0.01}. \quad (5.15)$$

We solve (5.14) in D8 using an initial discretization with 488 points. The gradient field obtained from the initial discretization is shown in Figure 5-7a. We denote by CA2 the adapted discretization that can be seen in Figure 5-7b with 849 points. The uniform discretizations CU3 and CU4 have 1579 and 1653 points, respectively. The discretization CU4 is shown in Figure 5-7c.

Table 5.4 shows the errors and the execution times of all discretizations. In the adapted discretization CA2 we achieved similar accuracy with a decrease of approximately 50% in the number of points and with a speedup of approximately 2 relative to the uniform discretization CU4.

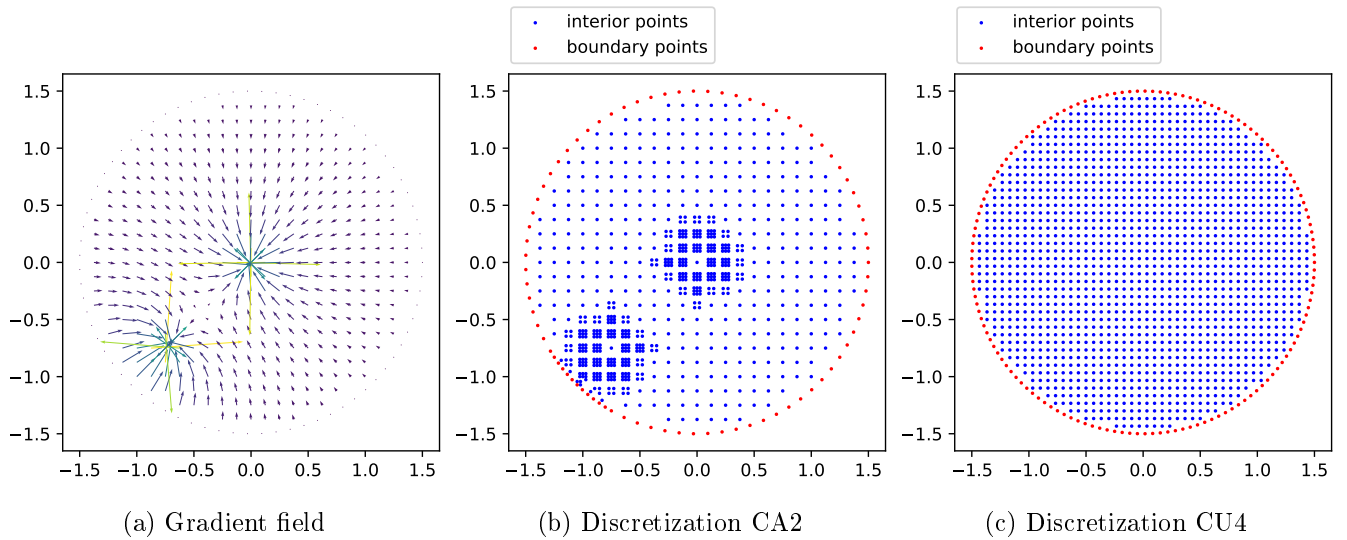


Figure 5-7: Gradients, adapted and uniform discretizations for Example 4.

Table 5.4: Global error and execution times for Example 4

Discretization	Total number of points (N)	Error (%)	Time (s)
CU3	1579	1.56	13.30
CU4	1653	1.00	14.20
CA2 (adapted)	849	1.19	7.75

Example 5

Consider the circular domain

$$D9 = \{(x, y) \in \mathbb{R}^2 | (x - 2.3)^2 + y^2 \leq 1.5^2\}$$

and the boundary value problem given by the following partial differential equation

$$\frac{\partial^2 U}{\partial x^2} - 2 \frac{\partial^2 U}{\partial x \partial y} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial U}{\partial x} - \frac{\partial U}{\partial y} = 0, \quad (5.16)$$

with exact solution

$$U(x, y) = \frac{1}{x + y}. \quad (5.17)$$

We solve (5.16) in $D9$ using an initial discretization with 488 points. The gradient field obtained from the initial discretization is shown in Figure 5-8a. We denote by CA3 the adapted discretization that can be seen in Figure 5-8b with 798 points. The uniform discretizations CU3 and CU6 have 1579 and 1506 points, respectively. The discretization CU3 is shown in Figure 5-8c.

Table 5.5 shows the results and the execution times of all discretizations. In the adapted discretization CA3 we achieved similar accuracy with a decrease of approximately 50% in the number of points and with a speedup of approximately 3 relative to the uniform discretization CU3.

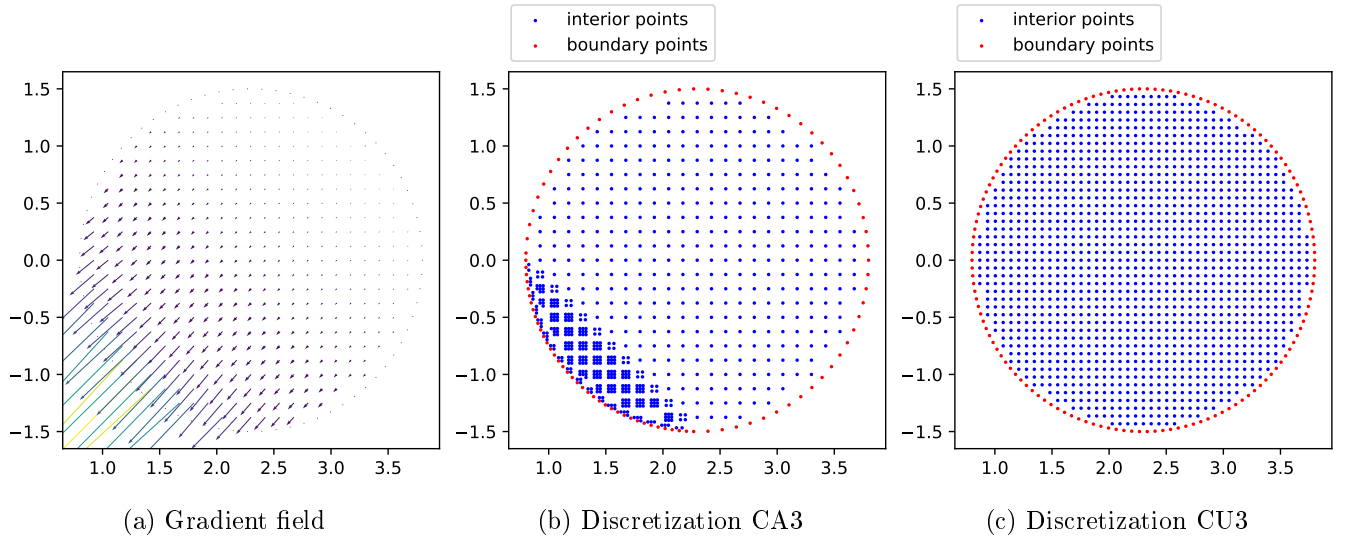


Figure 5-8: Gradients, adapted and uniform discretizations for Example 5.

Table 5.5: Global error and execution times for Example 5

Discretization	Total number of points (N)	Error (%)	Time (s)
CU6	1506	1.56e-1	10.13
CU3	1579	1.48e-1	13.30
CA3 (adapted)	798	1.54e-1	4.95

Example 6

Consider a circle with a lemniscate geometry inside it

$$D10 = C - L,$$

where $C = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1.5^2\}$ and $L = \{(x, y) \in \mathbb{R}^2 | (x^2 + y^2)^2 - (x^2 - y^2) < 0\}$.

Consider the boundary value problem given by the following partial differential equation

$$\frac{\partial^2 U}{\partial x^2} - \frac{\partial^2 U}{\partial y^2} = \frac{8x^2 - 8y^2}{(x^2 + y^2 + 0.01)^3} \quad (5.18)$$

with exact solution given in (5.13).

We solve (5.18) in the domain D10 using an initial discretization with 280 points. The gradient field obtained from the initial discretization is shown in Figure 5-9a. We denote by LA the adapted discretization that can be seen in Figure 5-9b with 493 points. The uniform discretizations LU1 and LU2 have 2032 and 2416 points, respectively. The discretization LU2 is shown in Figure 5-9c.

Table 5.6 shows the errors and the runtime of all discretizations. In the adapted discretization LA, we achieved a similar accuracy with a decrease of approximately 75% in the number of points and with a speedup of approximately 37 relative to the uniform discretization SU4.

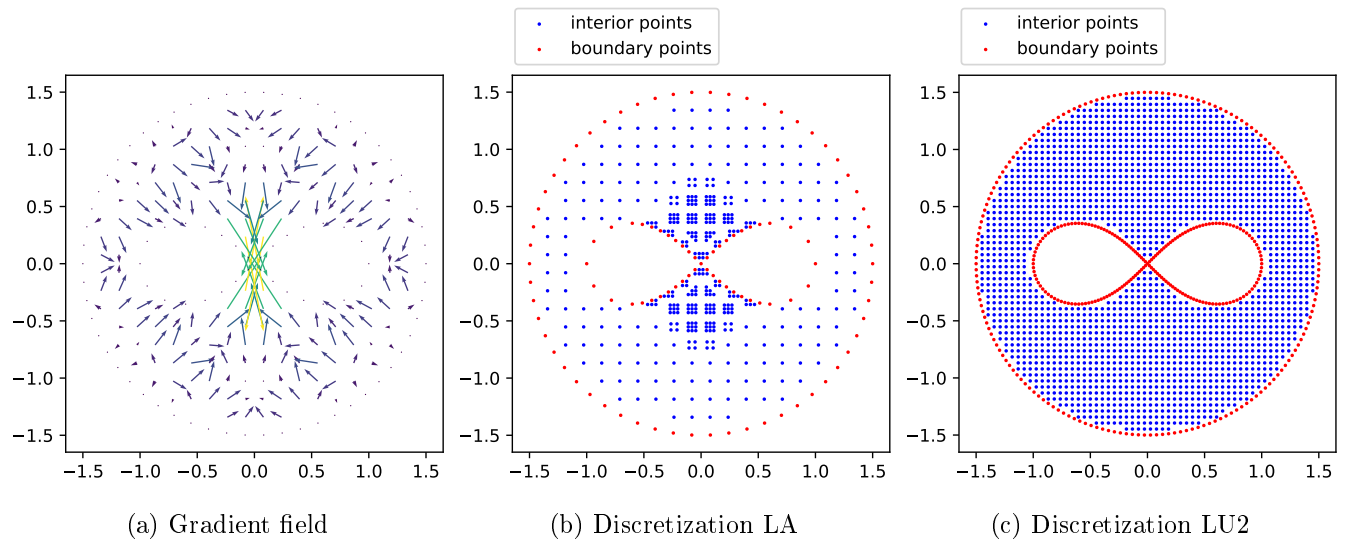


Figure 5-9: Gradients, adapted and uniform discretizations for Example 6.

Table 5.6: Global error and execution times for Example 6 with fourth-order approximations.

Discretization	Total number of points (N)	Error (%)	Time (s)
LU1	2032	4.24e-1	23.50
LU2	2416	1.71e-1	73.40
LA (Adapted)	493	1.19e-1	1.61

Additionally, we evaluated the number of corrected stars. Considering the LA and

LU1 discretizations, approximately 57% and 13% of the number of internal points have undergone a correction in their stars, respectively. Comparing the same discretizations but using the $\|\mathbf{x}_i - \mathbf{x}_0\|_2^{-2}$, $e^{-4\|\mathbf{x}_i - \mathbf{x}_0\|_2^2}$, and $e^{-9\|\mathbf{x}_i - \mathbf{x}_0\|_2^2}$ weighting functions, we obtain approximately 7%, 8%, and 5% of modified stars in the adapted discretization, respectively, and approximately 12%, 7%, and 7% in the uniform one, respectively. These results indicate that the type of discretization (adapted or uniform) is not a determining factor in the increase or decrease of the relative number of ill-conditioned stars.

5.2.2 Influence of the parameters η and β

Note that the parameter η influences the distance between points within the area of influence and, therefore, affects the accuracy of the derivatives. On the other hand, the parameter β controls the total amount of inserted points, the smaller the β , the more points will be added.

Figure 5-10 shows the global errors resulting from the variation of the parameter η in all adapted discretizations used in section 5.2.1. We varied η between 0.1 and 0.4 with a step size equal to 0.05. In general, the results indicate that intermediate values of η provide more accurate results. In this thesis, we have taken $\eta = 0.25$.

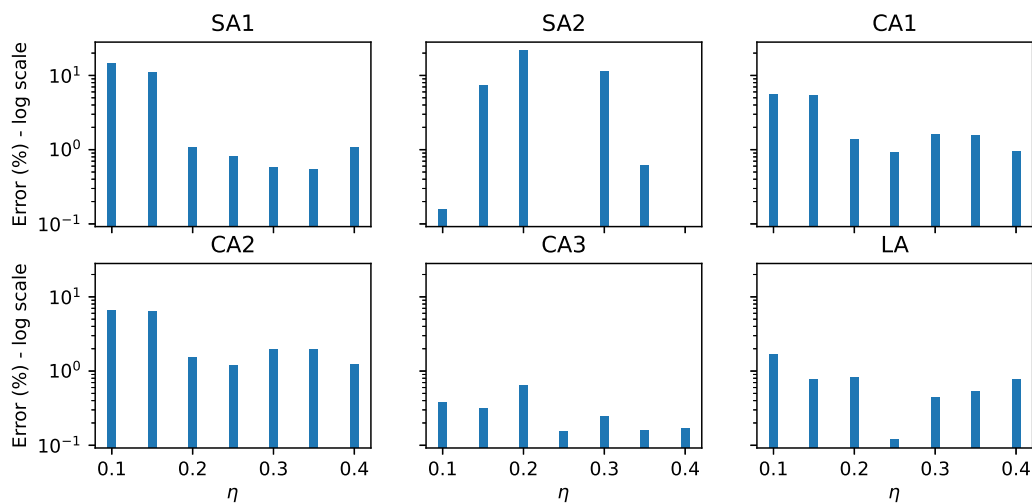


Figure 5-10: Relationship between the value of η and the global errors.

Figure 5-11 shows the global errors due to the variation of the parameter β . We varied β between 0 and 2 with a step size equal to 0.5. In general, values of β lower than 1 generate more accurate results. In this thesis, we have taken $\beta = 0.5$.

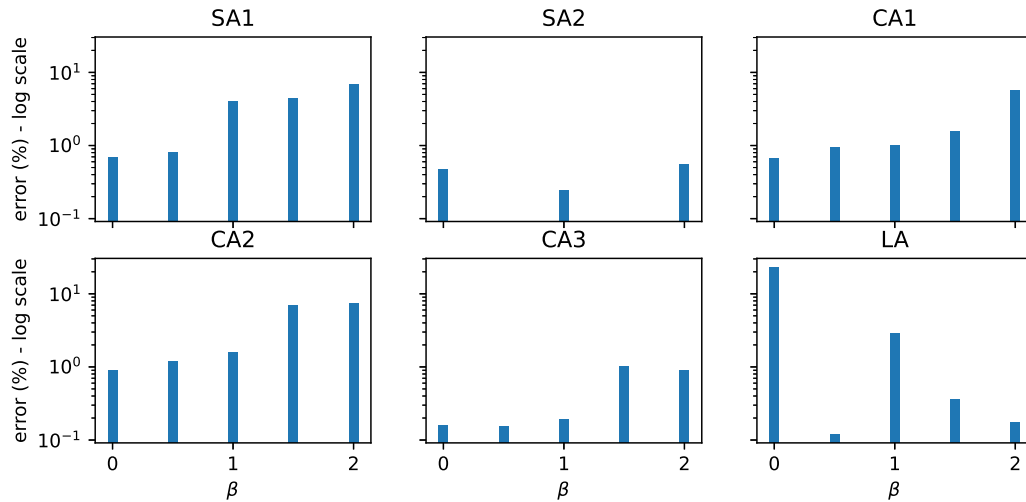


Figure 5-11: Relationship between the value of β and the global errors.

5.2.3 Influence of the weighting functions

Since weighting functions are expected to influence the accuracy of derivatives, we analyzed different weighting functions for Examples 1 and 3.

Table 5.7 shows the data for different weighting functions. In all cases, we used the same initial coarse discretizations and parameters α and β of section 5.2.1. The uniform discretizations considered satisfy that they have the smallest number of points which makes the error smaller than that obtained with the adapted discretization.

In these two examples, the potential weighting function with exponent -4 performed better in the adapted discretizations. Comparing the adapted and uniform discretizations, the results show the satisfactory performance of the adapted discretization regardless of the weighting functions used.

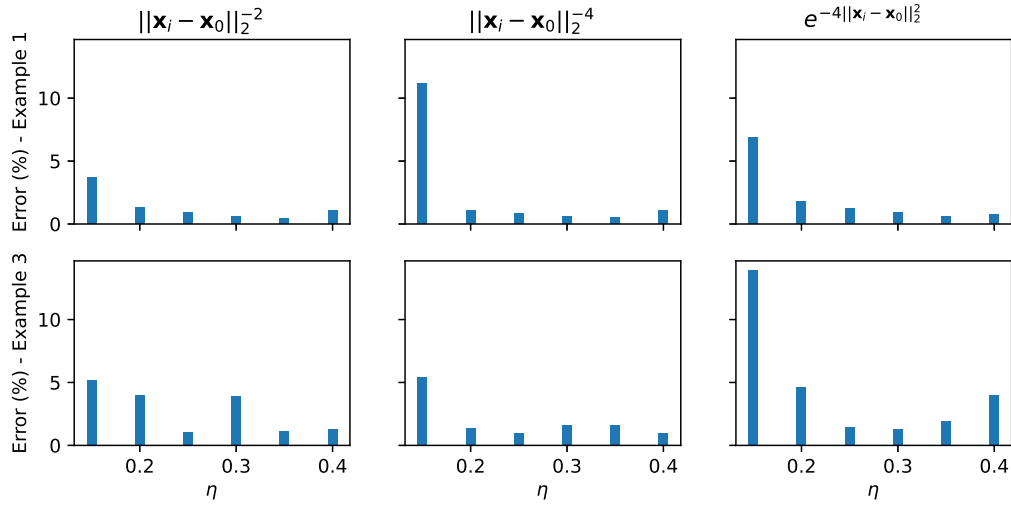


Figure 5-12: Relationship between the value of η , weighing functions, and global errors.

Table 5.7: Data for different weighing functions ($\eta = 0.25$, $\beta = 0.5$).

	Ex. 1 - adapted		Ex. 1 - uniform		Ex. 3 - adapted		Ex. 3 - Uniform	
Weighing functions	Error (%)	N	Error (%)	N	Error (%)	N	Error (%)	N
$\ \mathbf{x}_i - \mathbf{x}_0\ _2^{-2}$	9.05e-1	596	8.65e-1	1516	1.08	696	1.04	1653
$\ \mathbf{x}_i - \mathbf{x}_0\ _2^{-4}$	8.10e-1	596	7.93e-1	1681	0.94	696	6.17e-1	1799
$e^{-4\ \mathbf{x}_i - \mathbf{x}_0\ _2^2}$	1.25	596	1.15	1369	1.49	696	0.81	1653
$e^{-9\ \mathbf{x}_i - \mathbf{x}_0\ _2^2}$	1.25	596	1.15	1369	1.52	696	0.84	1653

Figure 5-12 shows the errors when we varied η and the weighing functions for Examples 1 and 3. It is evident that the accuracy of the results changes with different weighing functions, however, the form of the histograms does not vary significantly in Figure 5-12. Therefore, intermediate values of η are an appropriate choice regardless of the weighing functions tested.

5.2.4 Adapted discretizations with second-order approximations

We solve (5.8) and (5.10) using the second-order approximation and an initial discretization with 504 points.

We denote by SA3 and SA4 the adapted discretizations with 1134 and 1233 points, respectively (see Figures 5-13a and 5-13c). The uniform discretizations SU5, SU6, SU7, and SU8 have 2916, 3136, 5329 and 5476 points, respectively. The discretizations SU6 and SU8 are shown in Figures 5-13b and 5-13d, respectively.

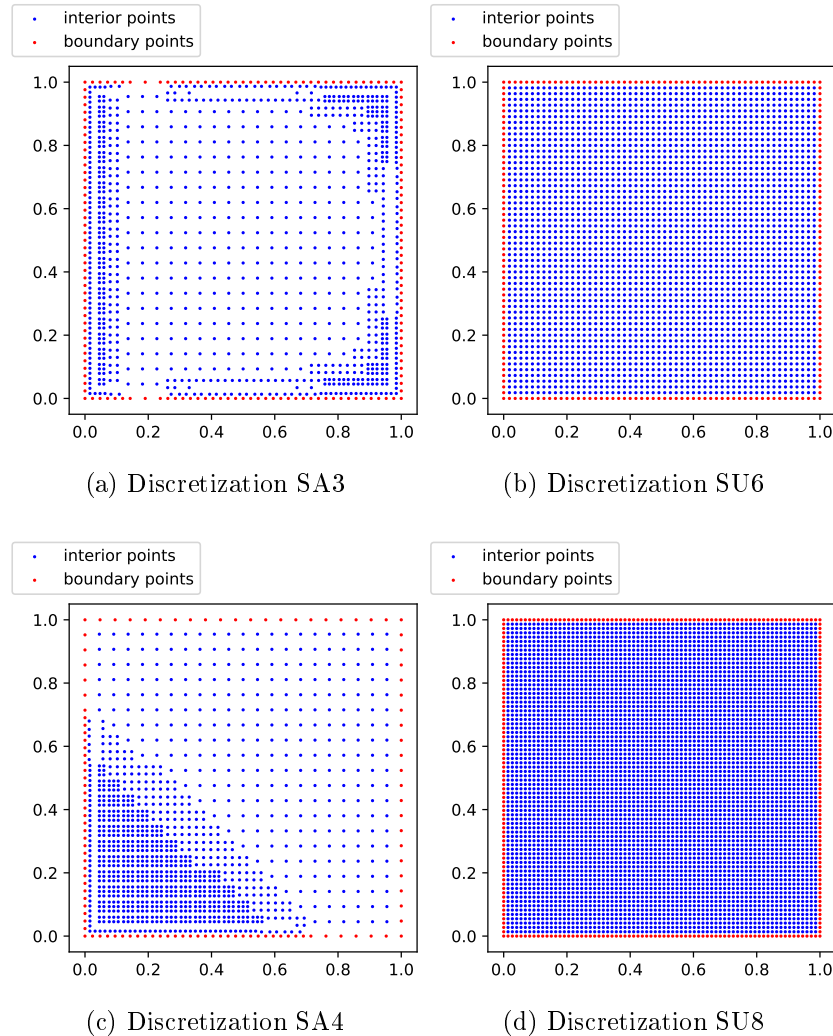


Figure 5-13: Adapted and uniform discretizations for Examples 1 and 2 with second-order approximations

Tables 5.8 and 5.9 show the results and execution times of all discretizations. In the adapted discretizations SA3 and SA4, we achieved a similar accuracy with a decrease of approximately 65% and 75%, respectively, in the number of points relative to the uniform discretizations SU6 and SU8, respectively. Comparing the

same discretizations, we get a speedups of approximately 5 and 15, respectively.

Table 5.8: Global error and execution times for Example 1 with second-order approximation

Discretization	Total number of points (N)	Error (%)	Time (s)
SU5	2916	1.38	6.86
SU6	3136	1.32	7.93
SA3 (adapted)	1134	1.32	1.48

Table 5.9: Global error and execution times for Example 2 with second-order approximation

Discretization	Total number of points (N)	Error (%)	Time (s)
SU7	5329	9.59e-1	22.55
SU8	5476	8.06e-2	25.85
SA4 (adapted)	1233	1.56e-1	1.75

Chapter 6

An h-adaptive method in 3D

6.1 Error indicator

In order to evaluate the performance of the adaptive procedure develop here, we will consider problems whose exact solution is known. The indicator error e_i was calculated by

$$e_i = \frac{|U_i - u_i|}{U_i}, \quad (6.1)$$

where U_i is the exact value of the solution at the point (x_i, y_i, z_i) and u_i is the approximate solution provided by the GFDM at the same point.

6.2 An adaptive procedure

At each adaptive step, we compute the value of the error indicator for each interior point using (6.1). Any point will be refined by adding new points if the corresponding value of the error indicator $e_i, i = 1, \dots, NI$, satisfies the following equation:

$$\bar{e} + \sigma_e < e_i, \quad (6.2)$$

where \bar{e} is the mean and σ_e is the standard deviation of the error indicators.

For each refined point, the number of points inserted will depend on the error

Table 6.1: Coordinates of the points to be added for each point (x_i, y_i, z_i) to be refined (e_i is the error indicator; \bar{e} and σ_e are the average and the standard deviation of the error indicators, respectively; Δ is the shortest distance between two points in all the domain).

Condition	Number of points added	Coordinates
$\bar{e} + \sigma_e < e_i \leq \bar{e} + 2\sigma_e$	8	$(x_i \pm \Delta/2, y_i \pm \Delta/2, z_i \pm \Delta/2)$
$e_i > \bar{e} + 2\sigma_e$	14	$(x_i \pm \Delta/2, y_i \pm \Delta/2, z_i \pm \Delta/2), (x_i \pm \Delta/2, 0, 0), (0, y_i \pm \Delta/2, 0),$ and $(0, 0, z_i \pm \Delta/2)$

indicator. Considering that Δ is the shortest distance between two points in all the domain, the proposed strategy is as follows:

- If $\bar{e} + \sigma_e < e_i \leq \bar{e} + 2\sigma_e$ we add 8 points, which are found at the vertices of a cube with edge Δ and geometric center at the point being refined.
- If $e_i > \bar{e} + 2\sigma_e$ we add 14 points, eight of them located as in the previous case, plus another six located in the centers of the faces of the cube.

Table 6.1 shows the coordinates of the points that are added in each case. Figure 6-1 illustrates the position of the added points in relation to the refined point.

At the end of each adaptive step, the algorithm will stop adding points if one of the following conditions is true:

1. for each interior point the inequality $e_i \leq \bar{e} + 2\sigma_e$ is satisfied;
2. the global errors in two successive refinements are greater than the global error in the current step. Suppose that the discretization in a step 4 has a global error equal to E_4 , in the following step 5 an error equal to E_5 , and in step 6 an error equal to E_6 . The stop occurs if $E_4 < E_5$ and $E_4 < E_6$. In this case, we accept as final discretization the one corresponding to step 4.

If a large number of steps occur, the domain can have a large concentration of points in a given region, so this may generate ill-conditioned stars. In this case, the procedure adaptive is inefficient and produces unsatisfactory results. Stopping criterion 2 gets us to abort before this problem occurs. Note that this criterion needs two

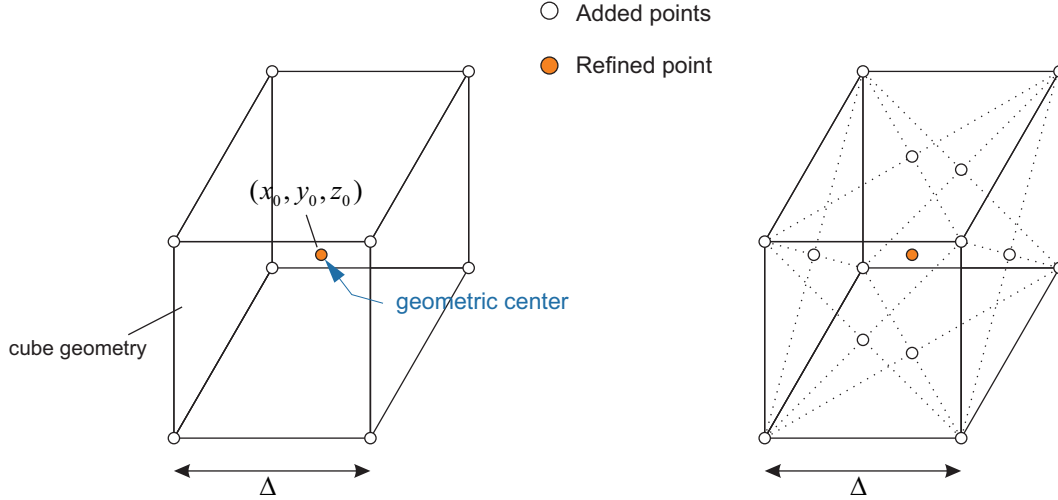


Figure 6-1: Position of the added points in relation to the refined point. On the left, 8 points are added; and on the right, 14 points are added.

extra steps before the stop, i.e, if the final discretization refers to step 'k' then the number of steps calculated was 'k+2'.

We impose a restriction for the distance between points in the h-adaptive method to avoid ill-conditioned stars, in such a way that if the distance between the new point and any point of the domain is smaller than $2\Delta/3$, then, we do not add new points.

6.3 Procedure for adding boundary points

Since we do not have error indicators for the boundary points, each boundary point searches the closest interior point. If this interior point was refined, then the respective boundary point also must be refined.

We explain how to insert points into the boundary with an example: consider a boundary point with coordinates (x_i, y_i, z_i) in the plane OXY, the four points inserted will have coordinates: $(x_i \pm 2\Delta, y_i, z_i)$ and $(x_i, y_i \pm 2\Delta, z_i)$; if the boundary point is in the plane OYZ, the points inserted will have coordinates: $(x_i, y_i \pm 2\Delta, z_i)$ and $(x_i, y_i, z_i \pm 2\Delta)$; and so on. Note that the points are inserted in a cross shape with center in (x_i, y_i, z_i) , another procedure should be adopted if the boundary is curved.

6.4 Examples

In all Examples, we compared our adaptive procedure with the one developed in [75]. In [75], for each refinement point the new points are added halfway between it and some points of the star. To do this, the algorithm selects the 8 furthest points of the star and adds a single point. However, in fourth-order approximations, more points per star are needed to improve the accuracy of the derivatives. Then, we selected the 14 farthest points from the star.

In all Examples, we used the weighting function $w_i = \|\mathbf{x}_i - \mathbf{x}_0\|_2^{-4}$, $i = 1, 2, \dots, m$, and stars with 90 points formed by the distance criterion.

The number of adaptive steps in all Examples does not include the initial solution.

The global errors are given by

$$\text{Error}(\%) = \frac{\sqrt{\frac{\sum_{i=1}^{\text{NI}} (U_i - u_i)^2}{\text{NI}}}}{u_{\max}} \cdot 100, \quad (6.3)$$

where U_i is the exact value of the solution at the point $(x_i, y_i) \in M \cap \Omega$, u_i is the approximate solution provided by the GFDM at the same point¹, and u_{\max} is the maximum value of the approximations given by the GFDM.

6.4.1 Example 1

Consider the domain given by

$$\text{D11} = \{(x, y, z) \in \mathbb{R}^3 \mid 0.01 \leq x, y, z \leq 1.01\},$$

taking an irregular initial discretization with 125 points (see Figure 6-2a).

¹For convenience, the numbering used here is global, as distinct from the local numbering u_i , $i = 0, 1, \dots, m$, used for each star in section 2.

We consider the boundary value problem given by the partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0, \quad (6.4)$$

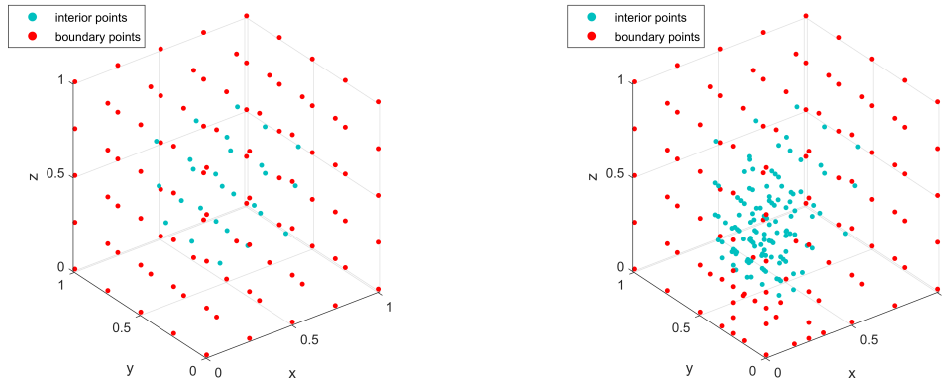
with exact solution $U = \frac{1}{\sqrt{x^2 + y^2 + z^2}}$. The boundary values are obtained from the exact solution.

We call Hex the algorithm developed in section 6.2 and Dist the distance algorithm previously explained in the introduction of the section 6.4. Table 6.2 shows the global errors, the final number of points, the number of adaptive steps and the time in the two algorithms. The algorithms stopped with final errors approximately equal, but we achieved a decrease of approximately 50% in the number of points, less number of steps, and a decrease of approximately 80% in the time compared to the Dist algorithm.

Table 6.2: Errors, number of points, number of adaptive steps, and time in each algorithm for Example 1.

Algorithm	Initial error (points)	Final error (points)	Number of steps	Time
Dist	3.426% (125)	0.286% (459)	7	68.25 s
Hex	3.426% (125)	0.280% (232)	5	13.40 s

Figure 6-2b shows the final discretization in the algorithm Hex with 232 points.



(a) Initial discretization with 125 points (b) Final discretization with 232 points

Figure 6-2: Discretizations in the adaptive algorithm for Example 1.

6.4.2 Example 2

Consider the same domain D11 and the initial discretization of Example 1.

We consider the boundary value problem given by the partial differential equation

$$\frac{\partial^2 U}{\partial x \partial y} + \frac{\partial^2 U}{\partial x \partial z} = 0, \quad (6.5)$$

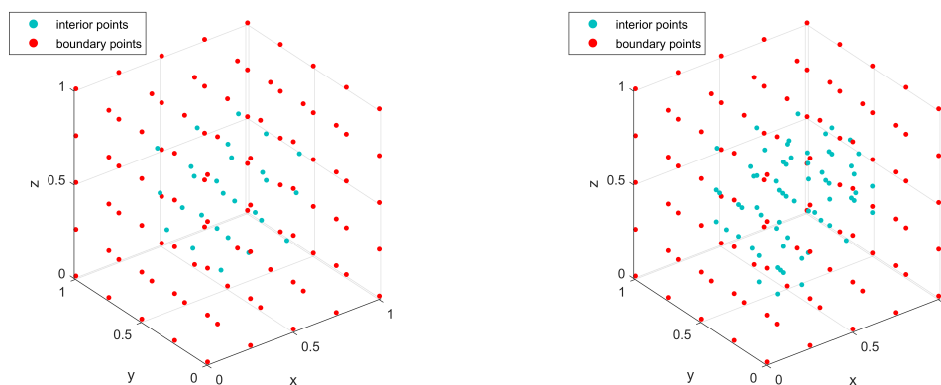
with exact solution $U = \frac{y - z}{1 + x}$. The boundary values are obtained from the exact solution.

Table 6.3 shows the global errors, the final number of points, the number of adaptive steps, and the time in the two algorithms. The algorithms stopped with the same number of steps and approximately the same time, but we achieved better accuracy and a decrease of approximately 20% in the number of points compared to the Dist algorithm.

Table 6.3: Errors, number of points, number of adaptive steps, and the time in each algorithm for Example 2.

Algorithm	Initial error (points)	Final error (points)	Number of steps	Time
Dist	8.27e-1% (125)	6.70e-2% (198)	4	7.65 s
Hex	8.27e-1% (125)	5.54e-2% (163)	4	6.13 s

Figure 6-3 shows the final discretization in the algorithm Hex with 163 points.



(a) Initial discretization with 125 points

(b) Final discretization with 163 points

Figure 6-3: Discretizations in the adaptive algorithm for Example 2.

6.4.3 Example 3

Consider the same domain D11 and the initial discretization of Example 1.

We consider the boundary value problem given by the partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 12e^{-2(x+y+z)}, \quad (6.6)$$

with exact solution $U = e^{-2(x+y+z)}$. The boundary values are obtained from the exact solution.

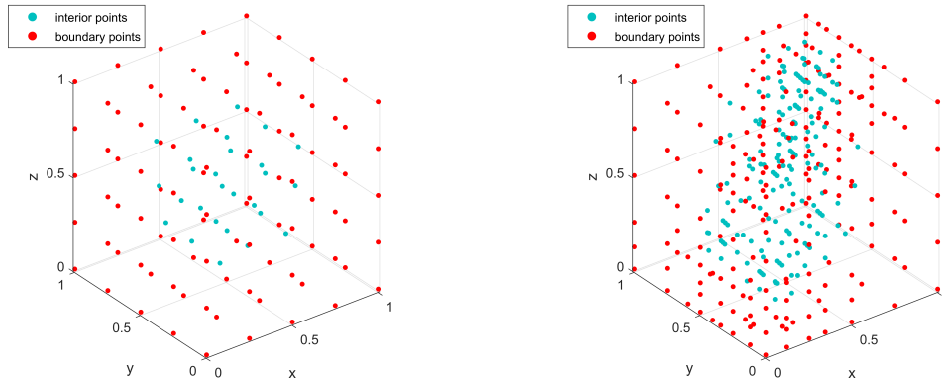
Table 6.4 shows the global errors, the final number of points, the number of adaptive steps, and the time in the two algorithms. We achieved better accuracy, fewer number of steps, and a decrease of approximately 10% and 50% in the number of

points and time, respectively, compared to the Dist algorithm.

Table 6.4: Errors, number of points, number of adaptive steps, and the time in each algorithm for Example 3.

Algorithm	Initial error (points)	Final error (points)	Number of steps	Time
Dist	1.42e-1% (125)	5.20e-2% (427)	6	41.04 s
Hex	1.42e-1% (125)	1.98e-2% (382)	5	21.89 s

Figure 6-4 shows the final discretization in the algorithm Hex with 382 points.



(a) Initial discretization with 125 points

(b) Final discretization with 382 points

Figure 6-4: Discretizations in the adaptive algorithm for Example 3.

6.4.4 Example 4

Consider a sphere with domain

$$D12 = \{(x, y, z) \in \mathbb{R}^3 | x^2 + y^2 + z^2 \leq 1\},$$

taking a initial discretization with 276 points (see Figure 6-5a).

The boundary value problem is given by the following partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + 2 \frac{\partial^2 U}{\partial z^2} = 0 \quad (6.7)$$

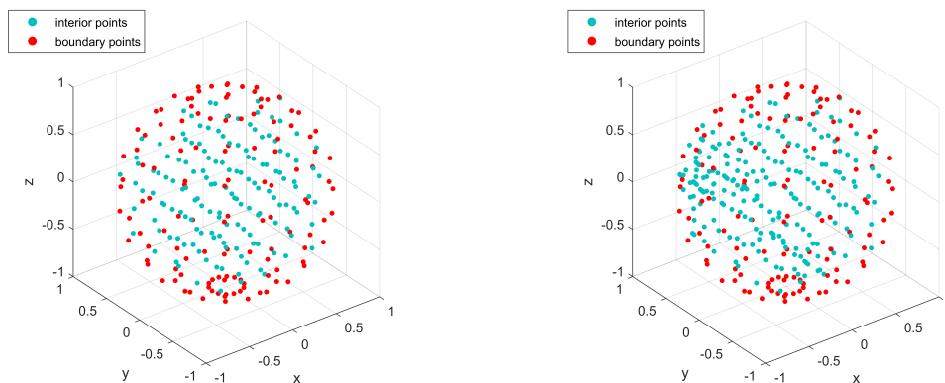
with Dirichlet boundary conditions and exact solution $U(x, y, z) = e^z \sin x \cos y$.

Table 6.5 shows the global errors, the final number of points, the number of adaptive steps, and the time in the two algorithms. We achieved better accuracy, fewer number of steps, and a decrease of approximately 20% in the number of points and time compared to the Dist algorithm.

Table 6.5: Errors, number of points, number of adaptive steps, and the time in each algorithm for Example 4.

Algorithm	Initial error (points)	Final error (points)	Number of steps	Time
Dist	2.89e-2% (276)	2.25e-2% (440)	4	19.30 s
Hex	2.89e-2% (276)	8.89e-3% (342)	3	15.11 s

Figure 6-5b shows the final discretization in the algorithm Hex with 342 points.



(a) Initial discretization with 276 points

(b) Final discretization with 342 points

Figure 6-5: Discretizations in the adaptive algorithm for Example 4.

6.4.5 Example 5

Consider the same domain D12 and the initial discretization of Example 4.

We consider a potential problem given by the partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = [4 - 4z^2 (x^2 + y^2)] \sin(1 + z^2) + 2(x^2 + y^2) \cos(1 + z^2) \quad (6.8)$$

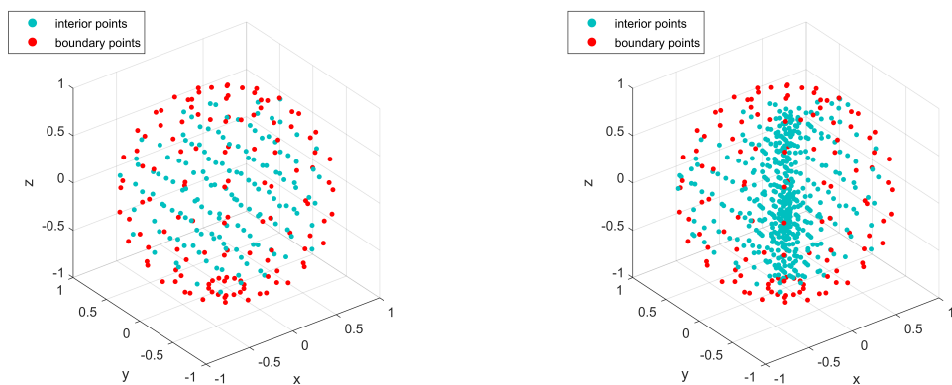
with Dirichlet boundary conditions and exact solution $U(x, y, z) = (x^2 + y^2) \sin(1 + z^2)$.

Table 6.6 shows the global errors, the final number of points, number of adaptive steps, and the time in the two algorithms. In this Example, the algorithm Dist failed to reduced the initial error, i.e, the addition of points did not provide better accuracy with respect to the error initial. On the other hand, we achieved a good performance with Hex.

Table 6.6: Errors, number of points, number of adaptive steps, and the time in each algorithm for Example 5.

Algorithm	Initial error (points)	Final error (points)	Number of steps	Time
Dist	4.86e-1% (276)	-	-	-
Hex	4.86e-1% (276)	1.74e-1% (620)	7	17.70 s

Figure 6-6 shows the final discretization in the algorithm Hex with 620 points.



(a) Initial discretization with 276 points

(b) Final discretization with 620 points

Figure 6-6: Discretizations in the adaptive algorithm for Example 5.

Chapter 7

Seismic response in onshore wind turbines

7.1 Partial differential equation of motion - beam bending

A model for Euler-Bernoulli transverse beam vibration [17], of variable cross-section, with elastic and homogenous material can be given by the fourth-order linear differential equation

$$\frac{\partial^2}{\partial x^2} \left(EI(x) \frac{\partial^2 \nu(x, t)}{\partial x^2} \right) + \frac{\partial}{\partial x} \left(Q \frac{\partial \nu(x, t)}{\partial x} \right) + \mu(x) \ddot{\nu}(x, t) = f_g(x, t). \quad (7.1)$$

Where $\nu(x, t)$ the transverse displacement, E the Young's modulus, $I(x)$ the cross-sectional moment of inertia, t is the time, the symbol (\cdot) denotes derivative with respect to time, $f_g(x, t)$ is the transversal force along the structure, $\mu(x)$ is the structural mass per unit length, and Q is the axial force along the structure. The parameters of (7.1) are represented in Figure 7-1.

Assuming Q constant and using the product rule for derivatives in (7.1) we get:

$$\left(E \frac{d^2 I(x)}{dx^2} + Q \right) \frac{\partial^2 \nu(x, t)}{\partial x^2} + 2E \frac{dI(x)}{dx} \frac{\partial^3 \nu(x, t)}{\partial x^3} + EI(x) \frac{\partial^4 \nu(x, t)}{\partial x^4} + \mu(x) \ddot{\nu}(x, t) = f_g(x, t) \quad (7.2)$$

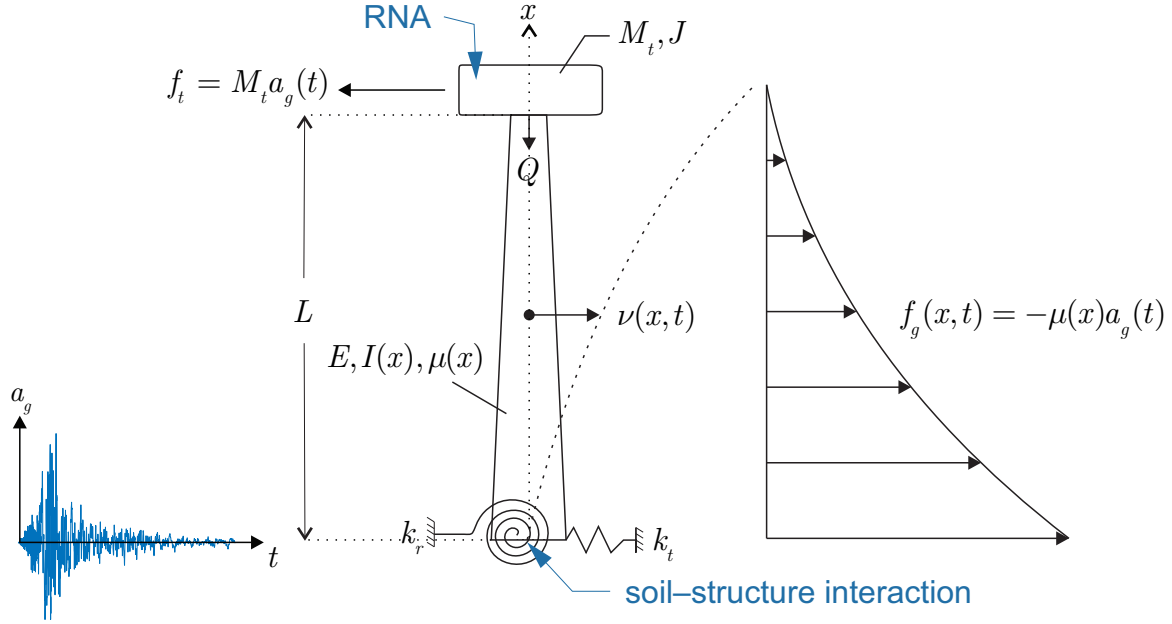


Figure 7-1: Euler-Bernoulli beam with a top mass and cross section variable subjected to a horizontal earthquake. The soil-structure interaction is modelled by two flexible springs.

The equation (7.2) can represent an idealized model of a wind turbine. The RNA (rotor-nacelle-assembly) of the wind turbine is idealized in the concentrated mass M_t - perfectly centered horizontally and without vertical eccentricity at the top of the structure - and in the rotary inertia J . The axial-force is given by $Q = -M_t \cdot g$, being g the gravitational acceleration. The soil-structure interaction is represented by the foundation lateral stiffness k_t and the foundation rocking stiffness k_r .

The value of f_g in (7.2) can be caused by various environmental effects, in this case, we considered the action of an earthquake. Let us consider as a_g the horizontal acceleration caused by an earthquake at the base of a wind turbine, then we have the relation $f_g(x, t) = -\mu(x) \cdot a_g(t)$, that way we transformed the acceleration at the base in effective earthquake force applied in the structure.

Due to presence of a concentrated mass M_t , the acceleration in the base generates a load point given by $f_t(t) = M_t \cdot a_g(t)$ (note that f_t and f_g must be in opposite directions).

The four boundary conditions of (7.2) are given by

$$EI(0)\frac{\partial^3\nu(0,t)}{\partial x^3} + Q\frac{\partial\nu(0,t)}{\partial x} + k_t\nu(0,t) = 0, \quad (7.3)$$

$$EI(0)\frac{\partial^2\nu(0,t)}{\partial x^2} - k_r\frac{\partial\nu(0,t)}{\partial x} = 0, \quad (7.4)$$

$$E\frac{dI(L)}{dx}\frac{\partial^2\nu(L,t)}{\partial x^2} + EI(L)\frac{\partial^3\nu(L,t)}{\partial x^3} + Q\frac{\partial\nu(L,t)}{\partial x} - M_t\ddot{\nu}(L,t) = f_t(t), \quad (7.5)$$

and

$$EI(L)\frac{\partial^2\nu(L,t)}{\partial x^2} + J\frac{\partial\ddot{\nu}(L,t)}{\partial x} = 0, \quad (7.6)$$

where the set of equations (7.3)–(7.6) represents the bending moment and the shear force at the base ($x = 0$) and at the top ($x = L$), being L the height of the tower.

As for the initial conditions of (7.2), we have

$$\nu(x,0) = \frac{\partial\nu(x,0)}{\partial x} = 0. \quad (7.7)$$

The solution of (7.2) must, of course, satisfy the prescribed boundary conditions (7.3)–(7.6) and the initial conditions (7.7).

7.2 Analysis of dynamic response

The domain is discretized considering NI internal points, two boundary points and two virtual points (see Figure 7-2). Adopting by convenience the notation $u \rightarrow \hat{\nu}$ in (2.15) and using the approximations of second-, third- and fourth-order into (7.2), we get

$$\begin{aligned} & \left(EI_0'' + Q \right) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{2,i} + \sum_{i=1}^m b_{2,i} \hat{\nu}_i \right) + 2EI_0' \left(-\hat{\nu}_0 \sum_{i=1}^m b_{3,i} + \sum_{i=1}^m b_{3,i} \hat{\nu}_i \right) + \\ & EI_0 \left(-\hat{\nu}_0 \sum_{i=1}^m b_{4,i} + \sum_{i=1}^m b_{4,i} \hat{\nu}_i \right) + \mu_0 \hat{a}_0 = f_0, \end{aligned} \quad (7.8)$$

where $f_0, I_0, \hat{\nu}_0, \mu_0$, and \hat{a}_0 (\hat{a} is the approximate value of \ddot{v}) are the values of the parameters $f_g, I, \hat{\nu}, \mu$, and \hat{a} calculated at in central point, respectively. Note that I_0' and I_0'' are the values of the derivatives of I at in central point.

Applying the approximations of first, second, and third-order of (2.15) to the boundary conditions (7.3)-(7.5), three additional equations are generated:

$$EI(0) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{3,i} + \sum_{i=1}^m b_{3,i} \hat{\nu}_i \right) + Q \left(-\hat{\nu}_0 \sum_{i=1}^m b_{1,i} + \sum_{i=1}^m b_{1,i} \hat{\nu}_i \right) + k_t \hat{\nu}_0 = 0, \quad (7.9)$$

$$EI(0) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{2,i} + \sum_{i=1}^m b_{2,i} \hat{\nu}_i \right) - k_r \left(-\hat{\nu}_0 \sum_{i=1}^m b_{1,i} + \sum_{i=1}^m b_{1,i} \hat{\nu}_i \right) = 0, \quad (7.10)$$

and

$$\begin{aligned} & EI'(L) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{2,i} + \sum_{i=1}^m b_{2,i} \hat{\nu}_i \right) + EI(L) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{3,i} + \sum_{i=1}^m b_{3,i} \hat{\nu}_i \right) \\ & + Q \left(-\hat{\nu}_0 \sum_{i=1}^m b_{1,i} + \sum_{i=1}^m b_{1,i} \hat{\nu}_i \right) - M_t \hat{a}_t = f_t, \end{aligned} \quad (7.11)$$

where \hat{a}_t is the approximate acceleration in at top of the tower ($x = L$).

Note that in (7.6) the derivative of the acceleration needs to be approximated. To apply (2.15) in (7.6), we adopted by convenience $u \rightarrow \hat{a}$ in the second term of (7.6), then

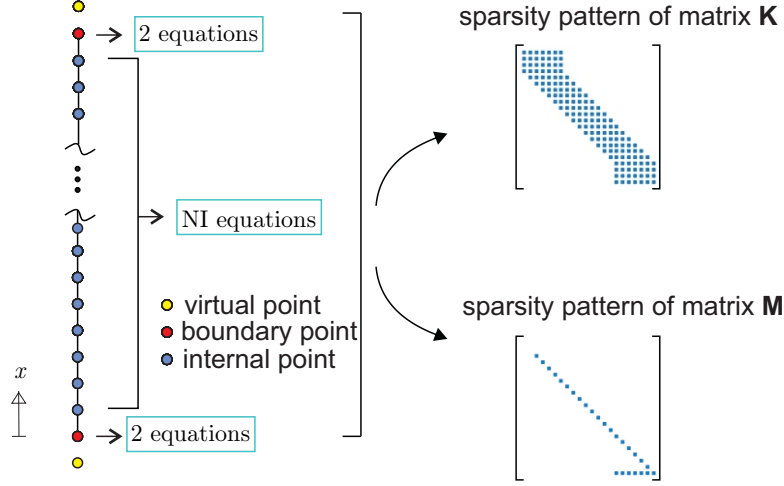


Figure 7-2: Discretization with the GFDM and sparsity pattern of matrices \mathbf{K} and \mathbf{M} .

$$EI(L) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{2,i} + \sum_{i=1}^m b_{2,i} \hat{\nu}_i \right) + J \left(-\hat{a}_0 \sum_{i=1}^m b_{1,i} + \sum_{i=1}^m b_{1,i} \hat{a}_i \right) = 0. \quad (7.12)$$

Applying equation (7.8) to each internal point of the domain together with the set (7.9)-(7.12), we obtain a system of equations with a stiffness matrix \mathbf{K} formed by the coefficients that multiply the displacements $\hat{\nu}$ and a mass matrix \mathbf{M} formed by the coefficients that multiply the accelerations \hat{a} . In compact form we have

$$\mathbf{K}\hat{\nu} + \mathbf{M}\hat{a} = \mathbf{F}\mathbf{g} \quad (7.13)$$

where $\hat{\nu}$ and \hat{a} are the vectors formed by the approximate displacements and accelerations in all points of the domain, respectively. $\mathbf{F}\mathbf{g}$ is the vector with the force applied at each point. The sparsity pattern of matrices \mathbf{K} and \mathbf{M} is shown in Figure (7-2).

A solution of (7.13) can be given by any appropriate integration method. We chose the Newmark Method [63], which is of easy programming and unconditionally stable (implicit method). Details of its computational implementation can be seen in Algorithm 4. We used the constant-average-acceleration method (also called trapezoidal rule).

Note that the initial vectors of displacement $\hat{\mathbf{v}}_0$, velocity $\hat{\mathbf{v}}_0$, and acceleration $\hat{\mathbf{a}}_0$ are considered equal to $\mathbf{0}$ to the seismic response.

Algorithm 4 A pseudocode to Newmark method.

Initialize $\hat{\mathbf{v}}_0$, $\hat{\mathbf{v}}_0$, $\hat{\mathbf{a}}_0$, and Δt . ▷ $\hat{\mathbf{v}}_0 = \hat{\mathbf{v}}_0 = \hat{\mathbf{a}}_0 = \mathbf{0}$ to the seismic response.

Calculate \mathbf{Fg}_i to each time discretization

Assemble the matrices \mathbf{K} and \mathbf{M}

Calculate c_1 and c_2 ▷ Equation (7.15)

$\mathbf{C} = c_1\mathbf{M} + c_2\mathbf{K}$ ▷ save \mathbf{C}

$\mathbf{K}' = \left(\mathbf{K} + \frac{4}{\Delta t^2}\mathbf{M} + \frac{2}{\Delta t}\mathbf{C} \right)^{-1}$ ▷ save the inverse

for $i = 1, 2, \dots$ number of time instants **do**

$\mathbf{Fg}' = \mathbf{Fg}_i + \mathbf{M} \left(\hat{\mathbf{a}}_{i-1} + \frac{4}{\Delta t}\hat{\mathbf{v}}_{i-1} + \frac{4}{\Delta t^2}\hat{\mathbf{v}}_{i-1} \right) + \mathbf{C} \left(\hat{\mathbf{v}}_{i-1} + \frac{2}{\Delta t}\hat{\mathbf{v}}_{i-1} \right)$

$\hat{\mathbf{v}}_i = \mathbf{K}' \cdot \mathbf{Fg}'$ ▷ multiplication of a matrix by a vector

$\hat{\mathbf{v}}_i = -\hat{\mathbf{v}}_{i-1} + \frac{2}{\Delta t}(\hat{\mathbf{v}}_i - \hat{\mathbf{v}}_{i-1})$

$\hat{\mathbf{a}}_i = -\hat{\mathbf{a}}_{i-1} + \frac{4}{\Delta t^2}(\hat{\mathbf{v}}_i - \hat{\mathbf{v}}_{i-1} - \hat{\mathbf{v}}_{i-1}\Delta t)$

end for

Another advantage of the Newmark Method is the possibility of introducing the damping matrix \mathbf{C} . The damping matrix can be calculated with the following linear combination:

$$\mathbf{C} = c_1\mathbf{K} + c_2\mathbf{M}. \quad (7.14)$$

The proportionality constants c_1 and c_2 are determined from the damping ratios of two natural vibration modes:

$$\begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} = 2 \frac{\omega_j \omega_i}{\omega_j^2 - \omega_i^2} \begin{bmatrix} \omega_j & -\omega_i \\ -1/\omega_j & 1/\omega_i \end{bmatrix} \begin{Bmatrix} \xi_i \\ \xi_j \end{Bmatrix}, \quad (7.15)$$

where the damping ratios ξ_i and ξ_j are related to i-th (ω_i) and the j-th (ω_j) vibration modes, respectively, that correspond to the first and to the least of the higher modes that contribute significantly to the dynamic response.

These vibration modes can be calculated by the GFDM using the procedure explained in the next section.

7.3 Procedure to calculate the natural vibration modes

Consider the same model of (7.1), but in terms of the natural frequency ω [17]:

$$\frac{d^2}{dx^2} \left(EI(x) \frac{d^2 \nu(x)}{dx^2} \right) + Q \left(\frac{d^2 \nu(x)}{dx^2} \right) - \mu(x) \omega^2 \nu(x) = 0. \quad (7.16)$$

Developing (7.16) with the product rule for derivatives, we get:

$$E \frac{d^2 I(x)}{dx^2} \frac{d^2 \nu(x)}{dx^2} + 2E \frac{dI(x)}{dx} \frac{d^3 \nu(x)}{dx^3} + EI(x) \frac{d^4 \nu(x)}{dx^4} + Q \frac{d^2 \nu(x)}{dx^2} - \mu(x) \omega^2 \nu(x) = 0, \quad (7.17)$$

which is a fourth-order linear differential equation, with variable coefficients, for free undamped vibrations.

The four boundary conditions associated to (7.17) can be given as follow

$$EI(0) \frac{d^2 \nu(0)}{dx^2} - k_r \frac{d\nu(0)}{dx} = 0, \quad (7.18)$$

$$EI(0) \frac{d^3 \nu(0)}{dx^3} + Q \frac{d\nu(0)}{dx} + k_t \nu = 0, \quad (7.19)$$

$$EI(L) \frac{d^2 \nu(L)}{dx^2} - J\omega^2 \frac{d\nu(L)}{dx} = 0, \quad (7.20)$$

$$E \frac{dI(L)}{dx} \frac{d^2 \nu(L)}{dx^2} + EI \frac{d^3 \nu(L)}{dx^3} + M_t \omega^2 \nu = 0, \quad (7.21)$$

where the equation set (7.18)–(7.21) represents the bending moment and the shear force at the base ($x = 0$) and at the top of the turbine ($x = L$).

Adopting again the notation $u \rightarrow \hat{\nu}$ in (2.15) and using the approximations of second-, third- and fourth-order into (7.17), we get:

$$\begin{aligned} & \left(EI_0'' + Q \right) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{2,j} + \sum_{i=1}^m b_{2,j} \hat{\nu}_i \right) + 2EI_0' \left(-\hat{\nu}_0 \sum_{i=1}^m b_{3,j} + \sum_{i=1}^m b_{3,j} \hat{\nu}_i \right) + \\ & EI_0 \left(-\hat{\nu}_0 \sum_{i=1}^m b_{4,j} + \sum_{i=1}^m b_{4,j} \hat{\nu}_i \right) - \mu_0 \omega \hat{\nu}_0 = 0. \end{aligned} \quad (7.22)$$

As for the boundary conditions, we use the approximations of first-, second-, and third-order of (2.15) in (7.18)-(7.21) which results in

$$EI(0) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{2,j} + \sum_{i=1}^m b_{2,j} \hat{\nu}_i \right) - k_r \left(-\hat{\nu}_0 \sum_{i=1}^m b_{1,j} + \sum_{i=1}^m b_{1,j} \hat{\nu}_i \right) = 0. \quad (7.23)$$

$$EI(0) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{3,j} + \sum_{i=1}^m b_{3,j} \hat{\nu}_i \right) + Q \left(-\hat{\nu}_0 \sum_{i=1}^m b_{1,j} + \sum_{i=1}^m b_{1,j} \hat{\nu}_i \right) + k_t \hat{\nu}_0 = 0, \quad (7.24)$$

$$EI(L) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{2,j} + \sum_{i=1}^m b_{2,j} \hat{\nu}_i \right) - J\omega^2 \left(-\hat{\nu}_0 \sum_{i=1}^m b_{1,j} + \sum_{i=1}^m b_{1,j} \hat{\nu}_i \right) = 0, \quad (7.25)$$

$$\begin{aligned} & EI'(L) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{2,j} + \sum_{i=1}^m b_{2,j} \hat{\nu}_i \right) + EI(L) \left(-\hat{\nu}_0 \sum_{i=1}^m b_{3,j} + \sum_{i=1}^m b_{3,j} \hat{\nu}_i \right) \\ & + M_t \omega^2 \hat{\nu}_0 = 0, \end{aligned} \quad (7.26)$$

A system of equations is formed by considering together (7.22)-(7.26)

$$\mathbf{K}\mathbf{u} + \omega\mathbf{u} = 0, \quad (7.27)$$

where ω is the frequency matrix. Note that the same stiffness matrix \mathbf{K} of section 7.2 is formed here. The equation (7.27) can be rewritten as follows

$$\mathbf{E}\mathbf{u} = 0, \quad (7.28)$$

where $\mathbf{E} = \mathbf{K} + \omega$. The eigenvalues can be found by expanding $\det(\mathbf{E}) = 0$ (non-trivial solution) and finding the roots of a polynomial of degree $2(N+2)$, which is called the characteristic equation (we recall that N is the total number of real points). Solving the characteristic equation yields $2(N+2)$ eigenvalues ω_k , $k = 1 \dots 2(N+2)$. We used the two smallest eigenvalues in (7.15) because they are the modes that contribute the most to the dynamic response of the turbine.

7.4 Turbine NREL

The National Renewable Energy Laboratory (NREL) offshore 5-MW baseline wind turbine has been used to establish the reference specifications for a number of research projects supported by the U.S. DOE's Wind and Hydropower Technologies Program [3].

The data of the NREL 5MW [47] are present in Table 7.1 . The density of 8,500 kg/m^3 was meant to be an increase over the typical steel value of 7,850 kg/m^3 to account for paint, bolts, welds, and flanges that are not accounted for in the tower thickness data. The radius and thickness of the tower were assumed to be linearly tapered from the base of the tower to its top.

The constant value of 5% in the damping ratio was based on an estimate of the seismic response for operating conditions in the fore-aft direction (direction parallel to the hub) [6, 80]. We used this damping ratio for all modes of vibration.

The values of lateral stiffness k_t and rocking stiffness k_r were obtained from a circular foundation in soil stratum over bedrock [48].

7.4.1 External force applied - earthquake

Earthquake chosen in the transient analysis was the Loma Prieta that struck the United States on October 18, 1989. The recording station is the 47381 [35], Figure

Table 7.1: Summary of Baseline Wind Turbine Properties (NREL- 5 MW)

Properties	Values
Rated power	5MW
Tower height	87300 mm
Tower bottom diameter	6000 mm
Tower top diameter	38700 mm
Tower wall thickness	27 - 19 mm
Tower density	8,500 kg/m^3
Mass of rotor-nacelle assembly	350 t
Tower Head moment of inertia about rotor-parallel axis through c.m.	43700 $t m^2$
Tower Head moment of inertia about lateral axis through c.m.	23500 $t m^2$
Damping ratio	5 %
Lateral stiffness of foundation	1.303e5 N/m
Rocking stiffness of foundation	2.607e5 N/rad

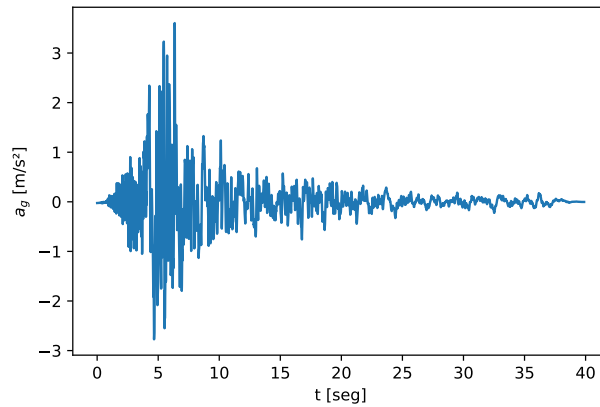


Figure 7-3: Accelerogram for the Loma Prieta earthquake.

7-3 shows the accelerogram in the horizontal direction of $PGA=0.37g$.

7.4.2 Result of dynamic response

As explained in section 7.2, to consider the damping ratio in the transient analysis it is necessary to calculate the natural frequencies of the structure. Table 7.2 shows the results of the natural frequencies in three different algorithms: (i) the algorithm Bmodes developed by Bir and Jonkman (2008) [47] uses a sophisticated approximation in finite element with an analytical linearization. The results are shown with 50

elements and each element with 15 degrees of freedom; (ii) A model based in the FEM using the software ABAQUS [1] with a uniform discretization of 20 elements of beam (B33 - cubic 2-node) to model the tower. A point mass and the rotational inertias are placed at the tip of the model representing the RNA; iii) The proposed strategy using the GFDM, which leads to the system (7.28). Here, we used the weighting function $w_i = \|\mathbf{x}_i - \mathbf{x}_0\|_2^{-6}$, $i = 1, 2, \dots, m$, the distance criterion, stars with six points, and a uniform discretization with 20 points.

Note that the modes are specified in terms of the tower fore-aft (FA) bending stiffness and side-side (SS) bending stiffness. In the first case, we used the moment of inertia with respect to the axis parallel to the RNA rotor, and in the second case, the moment of inertia about the lateral axis of the RNA. We have included the values of the Bmodes in Table 7.2 only to validate the proposed models in the FEM (ABAQUS) and in the GFDM. All the results are in good agreement.

Table 7.2: Modal frequencies to the Bmodes, FEM and GFDM in the turbine NREL

Mode Number	Mode Type	Frequency (Hz)		
		Bmodes	FEM	GFDM
1	1st SS	0.329	0.329	0.329
2	1st FA	0.332	0.332	0.332
3	2st SS	1.880	1.878	1.874
4	2st FA	2.243	2.278	2.277

From the values obtained in Table 7.2, the transient analysis with damping ratio can be evaluated. We adopted $\Delta t = 0.01$ in all results. In Figure 7-4, we compared the history of transversal displacement of the GFDM together with the FEM (ABAQUS) when we applied the seismic of Figure 7-3. The results are essentially identical and show the validity of the model proposed in the GFDM.

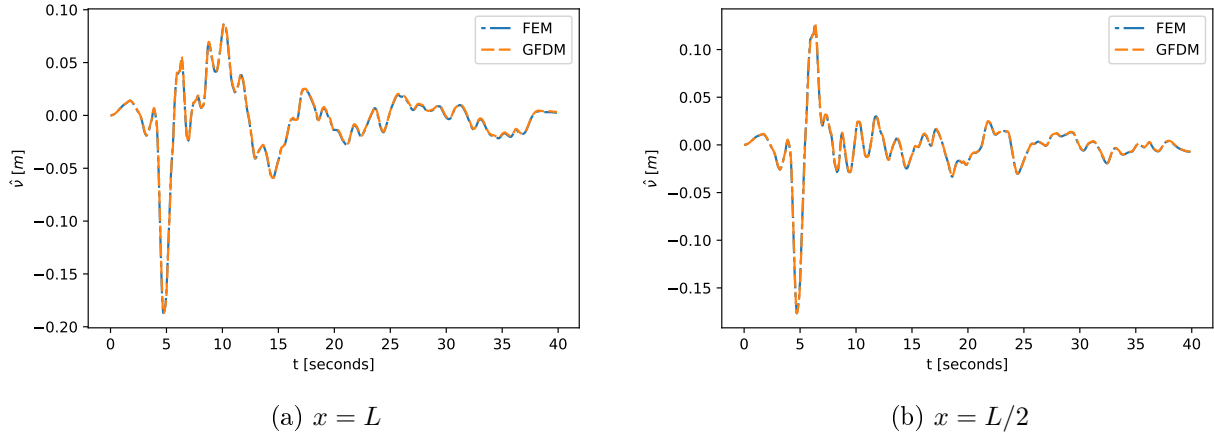


Figure 7-4: History of transversal displacement considering soil-structure interaction and damping ratio in the turbine NREL-5 MW.

7.5 Turbine Senvion MM92

The turbine Senvion MM92 is a variable speed wind turbine with a rated power of 2,050 kW. Like the entire MM series, it is based on the platform of the successful Senvion MD70/77.

Data of the Senvion MM92 [69] are present in Table 7.3. The MM92 tower consists of three sections assembled with two intermediate L flange connections. The variation of the radius along the structure is very similar to a linear variation, so we assumed that the radius of the tower is linearly tapered from the base of the tower to its top. In relation to the thickness, there is complexity because it has a non-linear variation throughout the tower, so we consider a constant thickness equal to 18 mm (approximate average thickness).

Table 7.3: Summary of Baseline Wind Turbine Properties (Senvion MM92- 2,050 kW)

Properties	Values
Rated power	2,050 kW
Tower height	76150 mm
Tower bottom diameter	4300 mm
Tower top diameter	2965 mm
Tower wall thickness	30 - 12 mm
Tower density	8,500 kg/m^3
Mass of rotor-nacelle assembly	108,800 t
Tower Head moment of inertia about rotor-parallel axis through c.m.	N/A
Tower Head moment of inertia about lateral axis through c.m.	N/A
Damping ratio	1.32 %
Lateral stiffness of foundation	1.44e1 MN/m
Rocking stiffness of foundation	3.20e3 MN/rad

The constant value of 1.32% in the damping ratio was based in an estimate of the viscous damping identified in [81]. We used this damping ratio for all modes of vibration.

The values of lateral stiffness k_t and rocking stiffness k_r were obtained from a circular foundation in compact sand [36].

Note that here the rotational inertia of the RNA was not provided by the manufacturer, so we consider in the numerical models $J = 0$.

7.5.1 External force applied - earthquake

The 1992 Cape Mendocino earthquakes occurred along the Lost Coast of Northern California and was chosen in transient analysis. The recording station is the 89005 [35], Figure 7-5 shows the accelelogram in the horizontal direction of PGA=1.50g.

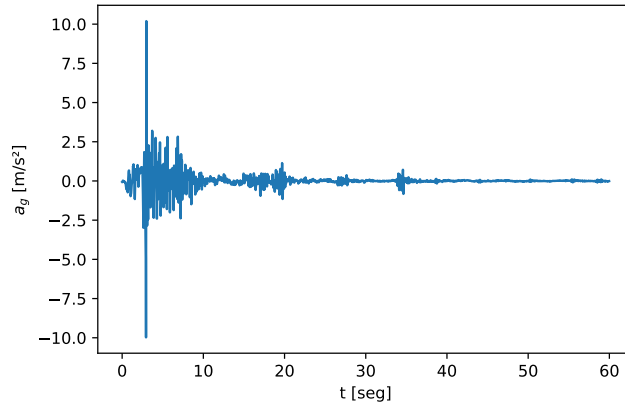


Figure 7-5: Accelerogram for the Cape Mendocino earthquake.

7.5.2 Result of dynamic response

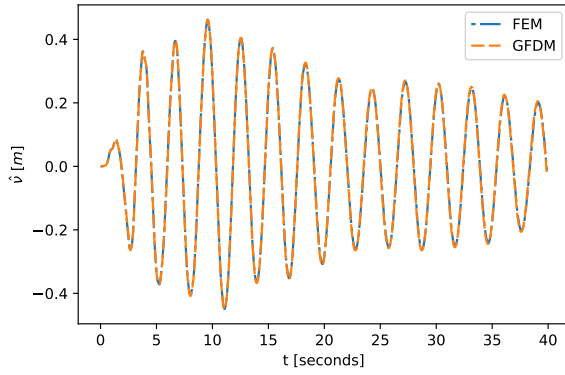
For reference, we used again the software ABAQUS to compare with the results of the GFDM. In both cases, the discretizations and the parameters related to the methods were the same as in section 7.4.

We show the natural frequencies in Table 7.4. Considering three decimal points, in the first mode the frequencies are in perfect agreement, while in the second mode there is a relative difference of approximately 7 %.

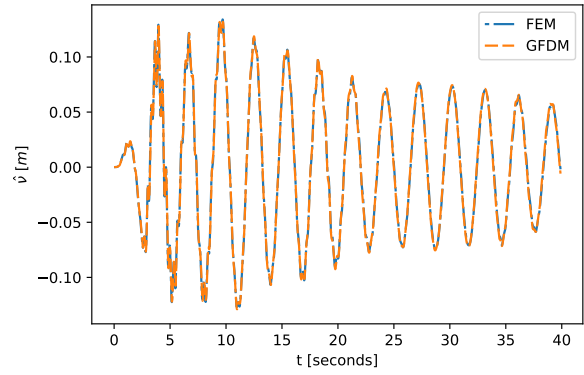
Table 7.4: Modal frequencies in the FEM and the GFDM to the turbine Senvion MM92.

Mode Number	Frequency [Hz]	
	FEM	GFDM
1	0.337	0.337
2	2.953	2.759

We adopted $\Delta t = 0.01$ in all transient analyses. In Figure 7-6 we compared the history of transversal displacement of the GFDM together with the FEM (ABAQUS) when we applied the seismic of Figure 7-5. The results are essentially identical and show again the validity of the model proposed in the GFDM.



(a) $x = L$



(b) $x = L/2$

Figure 7-6: History of transversal displacement considering soil-structure interaction and damping ratio in the turbine Senvion MM92.

Chapter 8

Conclusions and future lines of research

8.1 Conclusions

The main achievements obtained throughout the research related to the thesis completion are described below.

- **We have provided a number of points per star as a reference in third- and fourth-order approximations.**

For this purpose we have used irregular discretizations with a defined pattern and adaptable to any domain, but also irregular discretizations with a random distribution of points. In 2D, we have established 20 and 30 points as reference values for third- and fourth-order approximations, respectively, and in 3D, we have established 50 and 90 points as reference values for third- and fourth-order approximations, respectively.

- **We have reduced the computational cost in different stages of the calculation.**

For this we have introduced appropriate programming practices using higher-order approximations in the generalized finite difference method.

The algorithm includes efficient strategies with the assembly of sparse matrices and the parallel programming to distribute the work between processors and to obtain the stars and the coefficients of the derivatives.

However, even with the use of sparse matrix and parallel process, the algorithm can still have a high computational cost due to the loop needed to compute the derivatives for each point of the domain. So, we have presented a new version of the algorithm using a vector language, where this loop has been eliminated. The main idea is to transform 2D arrays into 3D arrays, where we used the additional dimension to store the matrices of each star in the domain. Broadcasting rules in Python facilitate the operations with 3D arrays.

Considering the steps of calculating the derivatives and assembling the system of equations, we have compared the computational times in the two versions (vectorized and non-vectorized). In 2D, the results show that the vectorized algorithm has been at least 150 times faster than the non-vectorized algorithm. In 3D, the vectorized algorithm has been at least 15 times faster.

The results have shown that vectorization of the derivatives provides much faster results compared to the parallel process of the derivatives. However, we indicate parallel processing in the formation of the stars, since it is easy to implement and vectorization is not possible at this stage.

- **We have presented a new approach to deal with ill-conditioned stars.**

The idea is to use the condition number of some regular stars as a threshold to detect stars that may be ill-conditioned and act only on these stars.

Specifically, we have used auxiliary regular stars to establish the tolerances of those stars with a minimum distance to the central point similar to the minimum distance to the central point of an auxiliary regular star. In order not to have to calculate an auxiliary regular star for each star in the domain, we have built a set of intervals, which contain the set of minimum distances, so that all those stars whose minimum distance is in the same interval are represented by the same auxiliary regular star. In addition, we have analyzed the influence of the

number of these intervals on the global error and on the number of modified stars, and the influence on the number of modified stars when applying different weighting functions.

To treat stars with a condition number greater than the established tolerance, we have followed a strategy with a maximum of two stages. Firstly, we changed the star formation criterion to that of the quadrant (2D) or of the octant (3D). And If that is not enough, secondly, we double the number of starting points in 2D or triple them in 3D.

In both 2D and 3D cases, the results of the application of this strategy show that simply using a number of points per star close to the minimum necessary, lower errors are obtained than using many more points per star. Furthermore, the number of auxiliary regular stars has been negligible and the percentage of ill-conditioned stars decreases rapidly with only a few points added to the star.

Above all, we highlight the good behavior of the condition number of the derivative matrix to detect ill-conditioned stars, regardless of the process followed to treat those stars. People working with the GFDM usually use a fixed number of points per star and the same selection criterion for all stars in the discretization. If the discretization is fairly regular, they usually use the distance criterion and a low number of points per star, while if the discretization is irregular, even if it is only irregular in some regions, they usually use the quadrant (octant) criterion and a much larger number of points. We would like to encourage the use of the proposed strategy because in this way everyone can continue to apply their own strategies for star formation, but now they will not have to worry about how the discretization is.

- **We have designed a strategy to generate a discretization adapted to the problem to be solved in a general way.**

Given a problem for which it is necessary to solve a differential equation in a domain employing the GFDM, the discretizations performed in such a domain generally have an approximately constant point density. Possibly, the most

widespread ways of discretizing are by means of mesh-based preprocessors or simply in a regular way, allowing for irregularities where this is not possible.

However, these forms of discretization do not allow to capture the characteristics of the problem and to take advantage of the benefits of GFDM. There are particular cases where the authors use discretizations with non-constant density in the domain to obtain higher accuracy using a smaller number of points. However, in such cases, there is prior knowledge.

We have proposed in this thesis to use discretizations adapted to the problem in general. To do this, we solve the problem in two stages. In the first stage, we solve the problem to compute the gradients using a regular coarse discretization. Once the gradients are calculated, we distribute the points according to the gradient values. Finally, we solve the problem considering the adapted discretization.

We have shown the performance of the proposed strategy for fourth-order approximations, but we have also shown some examples with second-order approximations where the results have been similar.

We have considered different weighting functions in the proposed strategy. The results show that the intermediate values chosen for η (expansion percentage of influence areas) are appropriate regardless of the tested weighting functions.

Furthermore, we would like to highlight two issues. On the one hand, the adapted discretizations provide the same accuracy as a regular discretization, but with a smaller number of points, with reductions above 50% in all our examples. In addition, the initial coarse discretization, that automatically generates the adapted discretization, has required between 9% and 27% of the points of the regular discretization.

On the other hand, the computational time required to solve the problem with these adapted discretizations, taking into account the whole process, is less, with reductions above 50% in all our examples.

- **We have developed a 3D adaptive algorithm with fourth-order ap-**

proximations on irregular initial discretizations.

The idea is to add points on the vertices and faces of a cube with a geometric center at the point that must be refined. The edge of the cube is the minimum distance between two points of the domain. This minimum distance is previously obtained from the star formation, which is a mandatory step in the method. Thus, there is no additional computational effort in forming the cubes.

We have established an adaptive algorithm that seeks to generate smooth transitions in different regions of the domain. The number of points inserted for each point refined depends on the error indicator values. The points with the largest error indicators are refined using 14 points. Points with indicator errors greater than the average and less than those previously selected are refined using 8 points.

We have compared the results with the algorithm of points added halfway. In all applications, we have achieved better accuracy with a decrease in the final number of points and time.

The adaptive algorithm developed here contributes to providing an efficient alternative in the three-dimensional scenario. On one hand, the algorithm does not require any mesh preprocessor. On the other hand, the results indicate a better performance compared to the strategy of inserting points halfway.

- **We have presented the evaluation of seismic responses in onshore wind turbines using the GFDM coupled with the Newmark method.**

The mathematical model is governed by a partial differential equation with variable coefficients (nonuniform cross-section). The problem is particularly challenging because it involves complex boundary conditions: rotary inertia and mass at the top of the tower, elastic supports at the base, and geometric nonlinearity provided by the RNA mass.

The strategy employed consists of obtaining equations of motion by the GFDM, i.e., we have obtained the stiffness and mass matrices after approximating the

derivatives by GFDM. Once these two matrices are obtained, we have applied the Newmark method to perform the time integration.

We have compared the history of transversal displacement with a model based on the FEM using the ABAQUS software. The results are essentially identical and show the validity of the model proposed in the GFDM.

We want to highlight a few points:

- We have included in the seismic response the Rayleigh damping, which is obtained by a linear combination of the mass and stiffness matrices.
- The verification of stability in the transient analysis is not necessary because the Newmark method is unconditionally stable. The choice of the stepsize integration is determined uniquely by accuracy considerations.
- In the transient analysis, only in the first interaction a system of equations has been solved. In the following interactions, only matrix-vector multiplication has been performed.
- The strategy employed here, where the mass and stiffness matrices are found using the GFDM, can be extended to the solution of other partial differential equations involving the time variable.

8.2 Future developments

This thesis investigated several aspects of high-performance programming using the generalized finite difference method. Topics that could be explored in the future are given below.

- Obtaining a hybrid method using the classical finite difference method with the GFDM. This combination can reduce the computational time in several steps of the code.
- The multiplication of 3D arrays in the vectorized algorithm can be accelerated with the NVIDIA's CUDA Python.

- Apply the discretizations adapted in equations with Neumann and Robin boundary conditions.
- Extend the adapted discretizations to 3D problems.
- Adopt an 3D adaptive procedure with the addition of points on curved boundary.
- Combine the multipoint technique with the high-order approximation. The idea is to further improve the accuracy of the higher-order approximations without increasing the number of points of the discretization.

Chapter 9

Conclusiones y futuras líneas de investigaciones

9.1 Conclusiones

Se describe a continuación los principales logros a lo largo de la investigación relacionados con la realización de la tesis.

- **Se ha propuesto un número de puntos por estrella como referencia en las aproximaciones de tercer y cuarto orden.**

Para ello hemos utilizado discretizaciones irregulares con un patrón definido y adaptable a cualquier dominio, pero también discretizaciones irregulares con una distribución aleatoria de puntos. En 2D, hemos establecido 20 y 30 puntos como valores de referencia para las aproximaciones de tercer y cuarto orden, respectivamente, y en 3D, hemos establecido 50 y 90 puntos como valores de referencia para las aproximaciones de tercer y cuarto orden, respectivamente.

- **Se ha reducido el coste computacional en diferentes etapas del cálculo.**

Para ello hemos introducido buenas prácticas de programación utilizando aproximaciones de orden superior en el método de diferencias finitas generalizado.

El algoritmo incluye estrategias eficientes con el ensamblaje de matrices disper-

sas y programación paralela para distribuir el trabajo entre los procesadores y obtener las estrellas y los coeficientes de las derivadas.

Sin embargo, incluso con el uso de matrices dispersas y procesos paralelos, el algoritmo puede seguir teniendo un alto coste computacional debido al bucle necesario para calcular las derivadas para cada punto del dominio. Por ello, hemos presentado una nueva versión del algoritmo utilizando un lenguaje vectorial, en el que se ha eliminado este bucle. La idea principal es transformar matrices 2D en matrices 3D, donde empleamos la dimensión adicional para almacenar las matrices de cada estrella del dominio. Las reglas de transmisión en Python facilitan las operaciones con matrices 3D.

Teniendo en cuenta los pasos necesarios para el cálculo de las derivadas y el ensamblaje del sistema de ecuaciones, hemos comparado los tiempos computacionales en las dos versiones (vectorizada y no vectorizada). En 2D, los resultados muestran que el algoritmo vectorizado ha sido al menos 150 veces más rápido que el algoritmo no vectorizado. En 3D, el algoritmo vectorizado ha sido al menos 15 veces más rápido.

Los resultados han demostrado que la vectorización de las derivadas proporciona resultados mucho más rápidos en comparación con el proceso paralelo de las derivadas. Sin embargo, aconsejamos el procesamiento paralelo en la formación de las estrellas, ya que es fácil de implementar y la vectorización no es posible en esta etapa.

- **Se ha presentado un nuevo enfoque para tratar las estrellas mal condicionadas.**

La idea es utilizar el número de condición de algunas estrellas regulares como umbral para detectar las estrellas que pueden estar mal condicionadas y actuar solamente sobre estas estrellas.

En concreto, hemos usado estrellas regulares auxiliares para establecer las tolerancias de aquellas estrellas con una distancia mínima al punto central similar a la distancia mínima al punto central de una estrella regular auxiliar. Para no

tener que calcular una estrella regular auxiliar para cada estrella del dominio, hemos construido un conjunto de intervalos, que contienen el conjunto de distancias mínimas, de forma que todas aquellas estrellas cuya distancia mínima está en el mismo intervalo están representadas por la misma estrella regular auxiliar. Además, hemos analizado la influencia del número de estos intervalos en el error global y en el número de estrellas modificadas, así como la influencia en el número de estrellas modificadas al aplicar diferentes funciones de ponderación.

Para tratar las estrellas con un número de condición superior a la tolerancia establecida, hemos seguido una estrategia con un máximo de dos etapas. En primer lugar, hemos cambiado el criterio de formación de estrellas por el del cuadrante (2D) o el del octante (3D). Y si eso no es suficiente, en segundo lugar, duplicamos el número de puntos de partida en 2D o los triplicamos en 3D.

Tanto en el caso 2D como en el 3D, los resultados de la aplicación de esta estrategia muestran que, simplemente utilizando un número de puntos por estrella cercano al mínimo necesario, se obtienen errores menores que utilizando muchos más puntos por estrella. Además, el número de estrellas regulares auxiliares ha sido despreciable y el porcentaje de estrellas mal condicionadas disminuye rápidamente con solamente unos pocos puntos añadidos a la estrella.

Sobre todo, destacamos el buen comportamiento del número de condición de la matriz de la derivada para detectar estrellas mal condicionadas, independientemente del proceso seguido para tratar esas estrellas. Las personas que trabajan con el MDFG suelen utilizar un número fijo de puntos por estrella y el mismo criterio de selección para todas las estrellas de la discretización. Si la discretización es bastante regular, suelen utilizar el criterio de la distancia y un número bajo de puntos por estrella, mientras que si la discretización es irregular, aunque solo lo sea en algunas regiones, suelen utilizar el criterio del cuadrante (octante) y un número mucho mayor de puntos. Nos gustaría animar

el uso de la estrategia propuesta porque de esta manera todos pueden seguir aplicando sus propias estrategias para la formación de estrellas, pero ahora no tendrán que preocuparse de cómo es la discretización.

- **Se ha desarrollado una estrategia para generar una discretización adaptada a problemas concretos.**

Dado un problema en el que es necesario resolver una ecuación diferencial en un dominio empleando el MDFG, las discretizaciones realizadas en dicho dominio suelen tener una densidad de puntos aproximadamente constante. Posiblemente, las formas más conocidas de discretización son mediante pre-procesadores basados en mallas o simplemente de forma regular, permitiendo irregularidades cuando esto no es posible.

Sin embargo, estas formas de discretización no permiten captar las características del problema y aprovechar las ventajas del MDFG. Existen casos particulares en el que los autores utilizan discretizaciones con densidad no constante en el dominio para obtener una mayor precisión utilizando un menor número de puntos pero se trata de casos en los que existe un conocimiento a priori.

Hemos propuesto en esta tesis utilizar discretizaciones adaptadas al problema en general. Para ello, resolvemos el problema en dos etapas. En la primera etapa, resolvemos el problema para calcular los gradientes utilizando una discretización regular gruesa. Una vez calculados los gradientes, distribuimos los puntos en función de los valores de los gradientes. Finalmente, resolvemos el problema considerando la discretización adaptada.

Hemos mostrado el rendimiento de la estrategia propuesta para aproximaciones de cuarto orden, pero también hemos mostrado algunos ejemplos con aproximaciones de segundo orden donde los resultados han sido similares.

Hemos considerado diferentes funciones de ponderación en la estrategia propuesta. Los resultados muestran que los valores intermedios elegidos para η (porcentaje de expansión de las áreas de influencia) son adecuados independientemente de las funciones de ponderación probadas.

Además, nos gustaría destacar dos cuestiones. Por un lado, las discretizaciones adaptadas proporcionan la misma precisión que una discretización regular, pero con un menor número de puntos, con reducciones superiores al 50% en todos los ejemplos tratados. Además, la discretización gruesa inicial, que genera automáticamente la discretización adaptada, ha requerido entre el 9% y el 27% de los puntos de la discretización regular.

Por otro lado, el tiempo computacional requerido para resolver el problema con estas discretizaciones adaptadas, teniendo en cuenta todo el proceso, es menor, con reducciones superiores al 50% en todos los ejemplos que se han considerado.

- **Se ha desarrollado un algoritmo adaptativo en 3D con aproximaciones de cuarto orden para discretizaciones iniciales irregulares.**

La idea es añadir puntos en los vértices y en las caras de un cubo con centro geométrico en el punto que debe ser refinado. La arista del cubo es la distancia mínima entre dos puntos del dominio. Esta distancia mínima se obtiene previamente en la formación de las estrellas, que es un paso obligatorio en el método. Así, no hay ningún esfuerzo computacional adicional en la formación de los cubos.

Hemos establecido un algoritmo adaptativo que busca generar transiciones suaves en diferentes regiones del dominio. El número de puntos insertados para cada punto refinado depende de los valores del indicador de error. Los puntos con los mayores indicadores de error se refinan con 14 puntos. Los puntos con errores de indicador mayores que la media y menores que los seleccionados previamente se refinan utilizando 8 puntos.

Hemos comparado los resultados con el algoritmo de puntos añadidos a media distancia. En todas las aplicaciones, hemos conseguido una mayor precisión con una disminución del número final de puntos y del tiempo.

El algoritmo adaptativo aquí desarrollado contribuye a proporcionar una alternativa eficiente en el escenario tridimensional. Por un lado, el algoritmo no requiere ningún pre-procesador de malla. Por otro lado, los resultados indican

un mejor rendimiento en comparación con la estrategia de insertar puntos a media distancia.

- **Se ha calculado la respuesta sísmica en aerogeneradores terrestres utilizando el MDFG acoplado con el método de Newmark**

El modelo matemático se rige por una ecuación diferencial parcial con coeficientes variables (sección transversal no uniforme). El problema es particularmente desafiante porque implica condiciones de contorno complejas: inercia y masa rotatoria en la parte superior de la torre, soportes elásticos en la base y no linealidad geométrica proporcionada por la masa del *RNA*.

La estrategia empleada consiste en resolver las ecuaciones de movimiento mediante el MDFG, es decir, hemos obtenido las matrices de rigidez y de masa tras aproximar las derivadas mediante el MDFG. Una vez obtenidas estas dos matrices, hemos aplicado el método de Newmark para realizar la integración temporal.

Hemos comparado el histórico del desplazamiento transversal con un modelo basado en el MEF utilizando el programa ABAQUS. Los resultados son esencialmente idénticos y muestran la validez del modelo propuesto en el MDFG.

Queremos destacar algunos puntos:

- Hemos incluido en la respuesta sísmica el amortiguamiento de Rayleigh, que se obtiene mediante una combinación lineal de las matrices de masa y rigidez.
- La verificación de la estabilidad en el análisis transitorio no es necesaria porque el método de Newmark es incondicionalmente estable. La elección del tamaño de integración de los pasos se determina únicamente por consideraciones de precisión.
- En el análisis transitorio, solo en la primera iteración se ha resuelto un sistema de ecuaciones. En las siguientes iteraciones, únicamente se ha ejecutado la multiplicación matriz-vector.

- La estrategia empleada aquí, donde las matrices de masa y rigidez se encuentran utilizando el MDFG, puede extenderse a la solución de otras ecuaciones diferenciales parciales que involucran la variable tiempo.

9.2 Desarrollos futuros

En esta tesis se han investigado varios aspectos de la programación de alto rendimiento mediante el método de las diferencias finitas generalizadas. A continuación se indican algunos temas que podrían explorarse en el futuro.

- Obtención de un método híbrido utilizando el método de las diferencias finitas clásico con el MDFG. Esta combinación puede reducir el tiempo de cálculo en varios pasos del código.
- La multiplicación de matrices 3D en el algoritmo vectorizado se puede acelerar con el CUDA Python de NVIDIA.
- Aplicar las discretizaciones adaptadas en ecuaciones con condiciones de contorno de Neumann y Robin.
- Extender las discretizaciones adaptadas a problemas en 3D.
- Adoptar un procedimiento adaptativo en 3D con la adición de puntos en contornos curvos.
- Combinar la técnica *multipunto* con la aproximación de alto orden. La idea es aumentar la precisión de las aproximaciones de alto orden sin aumentar el número de puntos de la discretización.

Bibliography

- [1] ABAQUS. Abaqus 6.11.
- [2] Sondipon Adhikari and S Bhattacharya. A general frequency adaptive framework for damped response analysis of wind turbines. *Soil Dynamics and Earthquake Engineering*, 143:106605, 2021.
- [3] P Agarwal and L Manuel. Simulation of offshore wind turbine response for extreme limit states. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 42681, pages 219–228, 2007.
- [4] SJ Ang, KS Yeo, CS Chew, and C Shu. A singular-value decomposition (svd)-based generalized finite difference (gfd) method for close-interaction moving boundary flow problems. *International journal for numerical methods in engineering*, 76(12):1892–1929, 2008.
- [5] Marino Arroyo and Michael Ortiz. Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods. *International journal for numerical methods in engineering*, 65(13):2167–2202, 2006.
- [6] ASCE and AWEA. Recommended practice for compliance of large land-based wind turbine support structures, 2011.
- [7] Juan José Benito, A García, L Gavete, M Negreanu, F Ureña, and Antonio Manuel Vargas. On the numerical solution to a parabolic-elliptic system with chemotactic and periodic terms using generalized finite differences. *Engineering Analysis with Boundary Elements*, 113:181–190, 2020.
- [8] Juan José Benito, F Ureña, and L Gavete. Influence of several factors in the generalized finite difference method. *Applied Mathematical Modelling*, 25(12):1039–1053, 2001.
- [9] Juan José Benito, F Urena, and L Gavete. Solving parabolic and hyperbolic equations by the generalized finite difference method. *Journal of computational and applied mathematics*, 209(2):208–233, 2007.
- [10] Juan José Benito, F Urena, L Gavete, and R Alvarez. An h-adaptive method in the generalized finite differences. *Computer methods in applied mechanics and engineering*, 192(5-6):735–759, 2003.

- [11] Juan José Benito, Francisco Ureña, L Gavete, and B Alonso. Application of the generalized finite difference method to improve the approximated solution of pdes. *Computer Modelling in Engineering & Sciences*, 38:39–58, 2009.
- [12] Juan José Benito, Francisco Ureña, Luis Gavete, and Beatriz Alonso. A posteriori error estimator and indicator in generalized finite differences. application to improve the approximated solution of elliptic pdes. *International Journal of Computer Mathematics*, 85(3-4):359–370, 2008.
- [13] Juan José Benito, Francisco Ureña, Miguel Ureña, Eduardo Saletе, and Luis Gavete. A new meshless approach to deal with interfaces in seismic problems. *Applied Mathematical Modelling*, 58:447–458, 2018.
- [14] Michael ZQ Chen, Zengmei Li, Haoyu Wang, and Yinlong Hu. Seismic response mitigation of a wind turbine via inerter-based structural control. *Bulletin of Earthquake Engineering*, pages 1–28, 2021.
- [15] Youping Chen, James D Lee, and Azim Eskandarian. *Meshless methods in solid mechanics*, volume 9. Springer, 2006.
- [16] Ni Chi-Mou. A quadrilateral finite difference plate element for nonlinear transient analysis of panels. *Computers & Structures*, 15(1):1–10, 1982.
- [17] RW Clough and J Penzien. *of Structures*. McGraw-Hill, 1975.
- [18] Lothar Collatz. *Numerische behandlung von differentialgleichungen*. Springer-Verlag, 1955.
- [19] Lothar Collatz. *The numerical treatment of differential equations*. Springer Science & Business Media, 2012.
- [20] François Cuvelier, Caroline Japhet, and Gilles Scarella. An efficient way to assemble finite element matrices in vector languages. *BIT Numerical Mathematics*, 56(3):833–864, 2016.
- [21] Chia-Ming Fan, Yu-Kai Huang, Po-Wei Li, and Chia-Lin Chiu. Application of the generalized finite-difference method to inverse biharmonic boundary-value problems. *Numerical Heat Transfer, Part B: Fundamentals*, 65(2):129–154, 2014.
- [22] Augusto César Albuquerque Ferreira and Paulo Marcelo Vieira Ribeiro. Reduced-order strategy for meshless solution of plate bending problems with the generalized finite difference method. *Latin American Journal of Solids and Structures*, 16(1):1–21, 2019.
- [23] George Elmer Forsythe and Wolfgang Richard Wasow. Finite-difference methods for partial differential equations. 1960.

- [24] Zhuo-Jia Fu, Ai-Lun Li, Chuanzeng Zhang, Chia-Ming Fan, and Xiao-Ying Zhuang. A localized meshless collocation method for bandgap calculation of anti-plane waves in 2d solid phononic crystals. *Engineering Analysis with Boundary Elements*, 119:162–182, 2020.
- [25] Zhuo-Jia Fu, Zhuo-Yu Xie, Shun-Ying Ji, Chia-Cheng Tsai, and Ai-Lun Li. Meshless generalized finite difference method for water wave interactions with multiple-bottom-seated-cylinder-array structures. *Ocean Engineering*, 195:106736, 2020.
- [26] Miguel A. García-March, Miguel Arevalillo-Herráez, Francisco R. Villatoro, Fernando Giménez, and Pedro Fernández de Córdoba. A generalized finite difference method using coatmêlec lattices. *Computer Physics Communications*, 180(7):1125–1133, 2009.
- [27] L Gavete, JJ Benito, and F Ureña. Generalized finite differences for solving 3d elliptic and parabolic equations. *Applied Mathematical Modelling*, 40(2):955–965, 2016.
- [28] L Gavete, ML Gavete, and Juan José Benito. Improvements of generalized finite difference method and comparison with other meshless method. *Applied Mathematical Modelling*, 27(10):831–847, 2003.
- [29] Luis Gavete, María Lucía Gavete, Francisco Ureña, and Juan José Benito. An approach to refinement of irregular clouds of points using generalized finite differences. *Mathematical Problems in Engineering*, 2015, 2015.
- [30] Luis Gavete, Francisco Ureña, Juan José Benito, and Eduardo Salet. A note on the dynamic analysis using the generalized finite difference method. *Journal of Computational and Applied Mathematics*, 252:132–147, 2013.
- [31] Luis Gavete, Francisco Ureña, Juan Jose Benito, Miguel Ureña, and Maria Lucia Gavete. Solving elliptical equations in 3d by means of an adaptive refinement in generalized finite differences. *Mathematical Problems in Engineering*, 2018, 2018.
- [32] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, 12 1977.
- [33] Yan Gu, Qingsong Hua, Chuanzeng Zhang, and Xiaoqiao He. The generalized finite difference method for long-time transient heat conduction in 3d anisotropic composite materials. *Applied Mathematical Modelling*, 71:316–330, 2019.
- [34] Yan Gu, Wenzhen Qu, Wen Chen, Lina Song, and Chuanzeng Zhang. The generalized finite difference method for long-time dynamic modeling of three-dimensional coupled thermoelasticity problems. *Journal of Computational Physics*, 384:42–59, 2019.

- [35] Hamid Haddadi, A Shakal, C Stephens, W Savage, M Huang, W Leith, and J Parrish. Center for engineering strong-motion data (cesmd). In *Proceedings of the 14th World Conference on Earthquake Engineering, Beijing, October*, pages 12–17, 2008.
- [36] Ma Hongwang. Seismic analysis for wind turbines including soil-structure interaction combining vertical and horizontal earthquake. In *15th World Conference on Earthquake Engineering, Lisbon, Portugal, 2012*.
- [37] Seyed Mahmoud Hosseini. Analysis of elastic wave propagation in a functionally graded thick hollow cylinder using a hybrid mesh-free method. *Engineering analysis with boundary elements*, 36(11):1536–1545, 2012.
- [38] Seyed Mahmoud Hosseini. Application of a hybrid mesh-free method based on generalized finite difference (gfd) method for natural frequency analysis of functionally graded nanocomposite cylinders reinforced by carbon nanotubes. *Computer Modeling in Engineering and Sciences-CMES*, 95(1):1–29, 2013.
- [39] Wen Hu, Yan Gu, and Chia-Ming Fan. A meshless collocation scheme for inverse heat conduction problem in three-dimensional functionally graded materials. *Engineering Analysis with Boundary Elements*, 114:1–7, 2020.
- [40] I Jaworska. *The multipoint meshless finite difference method for analysis of boundary value problems of mechanics*. PhD thesis, Ph. D. Thesis, Cracow University of Technology, Cracow, 2009.
- [41] Irena Jaworska and Janusz Orkisz. Higher order multipoint method—from collatz to meshless fdm. *Engineering Analysis with Boundary Elements*, 50:341–351, 2015.
- [42] Irena Jaworska and Janusz Orkisz. On nonlinear analysis by the multipoint meshless fdm. *Engineering Analysis with Boundary Elements*, 92:231–243, 2018. Improved Localized and Hybrid Meshless Methods - Part 1.
- [43] Paul S Jensen. Finite difference techniques for variable grids. *Computers & Structures*, 2(1-2):17–29, 1972.
- [44] Wenyu Jiang, Cheng Lin, and Min Sun. Seismic responses of monopile-supported offshore wind turbines in soft clays under scoured conditions. *Soil Dynamics and Earthquake Engineering*, 142:106549, 2021.
- [45] Xin Jin, Hua Liu, and Wenbin Ju. Wind turbine seismic load analysis based on numerical calculation. *Strojniski Vestnik/Journal of Mechanical Engineering*, 60(10), 2014.
- [46] E Jones, T Oliphant, and P Peterson. Scipy community, 2001. scipy: Open source scientific tools for python.

- [47] Jason Jonkman, Sandy Butterfield, Walter Musial, and George Scott. Definition of a 5-mw reference wind turbine for offshore system development. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2009.
- [48] Eduardo Kausel. Forced vibrations of circular foundations on layered media. *MIT research report*, 1974.
- [49] Fuat Korkut, Turgut Tokdemir, and Yalçın Mengi. The use of generalized finite difference method in perfectly matched layer analysis. *Applied Mathematical Modelling*, 60:127–144, 2018.
- [50] Jun Lei, Yanjie Xu, Yan Gu, and Chia-Ming Fan. The generalized finite difference method for in-plane crack problems. *Engineering Analysis with Boundary Elements*, 98:147–156, 2019.
- [51] Po-Wei Li, Zhuo-Jia Fu, Yan Gu, and Lina Song. The generalized finite difference method for the inverse cauchy problem in two-dimensional isotropic linear elasticity. *International Journal of Solids and Structures*, 174:69–84, 2019.
- [52] Tadeusz Liszka and Janusz Orkisz. The finite difference method at arbitrary irregular grids and its application in applied mechanics. *Computers & Structures*, 11(1-2):83–95, 1980.
- [53] T.J. Liszka, C.A.M. Duarte, and W.W. Tworzydło. hp-meshless cloud method. *Computer Methods in Applied Mechanics and Engineering*, 139(1):263–288, 1996.
- [54] Gui-Rong Liu and YuanTong Gu. A point interpolation method for two-dimensional solids. *International Journal for Numerical Methods in Engineering*, 50(4):937–951, 2001.
- [55] Wing Kam Liu, Sukky Jun, and Yi Fei Zhang. Reproducing kernel particle methods. *International journal for numerical methods in fluids*, 20(8-9):1081–1106, 1995.
- [56] YY Lu, T Belytschko, and Lu Gu. A new implementation of the element free galerkin method. *Computer methods in applied mechanics and engineering*, 113(3-4):397–414, 1994.
- [57] Hasan Malaeke and Hamid Moeenfarid. Analytical modeling of large amplitude free vibration of non-uniform beams carrying a both transversely and axially eccentric tip mass. *Journal of Sound and Vibration*, 366:211–229, 2016.
- [58] Cristina Medina, Guillermo M Álamo, and Román Quevedo-Reina. Evolution of the seismic response of monopile-supported offshore wind turbines of increasing size from 5 to 15 mw including dynamic soil-structure interaction. *Journal of Marine Science and Engineering*, 9(11):1285, 2021.

- [59] Sławomir Milewski. Selected computational aspects of the meshless finite difference method. *Numerical Algorithms*, 63(1):107–126, 2012.
- [60] Sławomir Milewski. Development of simple effective cloud of nodes and triangular mesh generators for meshless and element-based analyses-implementation in matlab. *Computer Assisted Methods in Engineering and Science*, 24(3):157–180, 2018.
- [61] Sławomir Milewski and Roman Putanowicz. Higher order meshless schemes applied to the finite element method in elliptic problems. *Computers & Mathematics with Applications*, 77(3):779–802, 2019.
- [62] Bernard Nayroles, Gilbert Touzot, and P38504470764 Villon. Generalizing the finite element method: diffuse approximation and diffuse elements. *Computational mechanics*, 10(5):307–318, 1992.
- [63] Nathan M Newmark and Emilio Rosenblueth. Fundamentals of earthquake engineering. prentice-hall, inc. *Englewood Cliffs, New Jersey*, 1971.
- [64] E. Oñate, S. Idelsohn, O.C. Zienkiewicz, R.L. Taylor, and C. Sacco. A stabilized finite point method for analysis of fluid mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 139(1):315–346, 1996.
- [65] J Orkisz. Finite difference method. *Handbook of Computational Solid Mechanics*, pages 336–432, 1998.
- [66] Janusz Orkisz and Sławomir Milewski. A’posteriori error estimation based on higher order approximation in the meshless finite difference method. In *Meshfree Methods for Partial Differential Equations IV*, pages 189–213. Springer, 2008.
- [67] Nicholas Perrone and Robert Kao. A general finite difference method for arbitrary meshes. *Computers & Structures*, 5(1):45–57, 1975.
- [68] Python. multiprocessing — process-based parallelism, 2021.
- [69] Senvion SE. Senvion mm92 - product description. Technical report, Senvion SE, Hamburg (Germany), 2015.
- [70] Pratik Suchde and Joerg Kuhnert. A meshfree generalized finite difference method for surface pdes. *Computers & Mathematics with Applications*, 78(8):2789–2805, 2019.
- [71] Francesca Taddei, Marco Schauer, and Lisanne Meinerzhagen. A practical soil-structure interaction model for a wind turbine subjected to seismic loads and emergency shutdown. *Procedia engineering*, 199:2433–2438, 2017.
- [72] AA Tseng and SX Gu. A finite difference scheme with arbitrary mesh systems for solving high-order partial differential equations. *Computers & structures*, 31(3):319–328, 1989.

- [73] A.A. Tseng and S.X. Gu. A finite difference scheme with arbitrary mesh systems for solving high-order partial differential equations. *Computers & Structures*, 31(3):319–328, 1989.
- [74] M. Ureña. *Método de las diferencias finitas generalizadas en tres dimensiones*. PhD thesis, Universidad Nacional de Educación a Distancia, 2014.
- [75] F Ureña, JJ Benito, R Alvarez, and L Gavete. Computational error approximation and h-adaptive algorithm for the 3-d generalized finite difference method. *International Journal for Computational Methods in Engineering Science and Mechanics*, 6(1):31–39, 2005.
- [76] F Ureña, L Gavete, A García, Juan José Benito, and Antonio Manuel Vargas. Solving second order non-linear hyperbolic pdes using generalized finite difference method (gfdm). *Journal of Computational and Applied Mathematics*, 363:1–21, 2020.
- [77] Francisco Ureña, Juan José Benito, Eduardo Saletе, and Luis Gavete. A note on the application of the generalized finite difference method to seismic wave propagation in 2d. *Journal of Computational and Applied Mathematics*, 236(12):3016–3025, 2012.
- [78] Miguel Ureña, Juan José Benito, Francisco Ureña, Ángel García, Luis Gavete, and Luis Benito. Adaptive strategies to improve the application of the generalized finite differences method in 2d and 3d. *Mathematical Methods in the Applied Sciences*, 41(17):7115–7129, 2018.
- [79] Miguel Ureña, Juan José Benito, Francisco Ureña, Eduardo Saletе, and Luis Gavete. Application of generalised finite differences method to reflection and transmission problems in seismic sh waves propagation. *Mathematical Methods in the Applied Sciences*, 41(6):2328–2339, 2018.
- [80] V Valamanesh and AT Myers. Aerodynamic damping and seismic response of horizontal axis wind turbine towers. *Journal of Structural Engineering*, 140(11):04014090, 2014.
- [81] Milan Veljkovic, Christine Heistermann, Wylliam Husson, Marouene Limam, M Feldmann, J Naumes, D Pak, T Faber, M Klose, KU Fruhner, et al. *High-strength tower in steel for wind turbines (HISTWIN)*. European Commission Joint Research Centre, 2012.
- [82] Zhaoyang Wang and HongGuang Sun. Generalized finite difference method with irregular mesh for a class of three-dimensional variable-order time-fractional advection-diffusion equations. *Engineering Analysis with Boundary Elements*, 132:345–355, 2021.

- [83] Jianguo Wei, Song Wang, Qingzhi Hou, and Jianwu Dang. Generalized finite difference time domain method and its application to acoustics. *Mathematical Problems in Engineering*, 2015, 2015.
- [84] MJ Wyatt, Taylor, G Davies, and C snell. A new difference based finite element method. *Proceedings of the Institution of Civil Engineers*, 59(3):395–409, 1975.
- [85] Qiang Xi, Zhuojia Fu, Yudong Li, and He Huang. A hybrid gfdm–sbm solver for acoustic radiation and propagation of thin plate structure under shallow sea environment. *Journal of Theoretical and Computational Acoustics*, 28(02):2050008, 2020.
- [86] Mi Zhao, Zhidong Gao, Piguang Wang, and Xiuli Du. Response spectrum method for seismic analysis of monopile offshore wind turbine. *Soil Dynamics and Earthquake Engineering*, 136:106212, 2020.
- [87] X Zhao and P Maisser. Seismic response analysis of wind turbine towers including soil-structure interaction. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 220(1):53–61, 2006.