Iterative solvers for RBF-FD discretized flow problems

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation

von
Willi Leinen

aus
Trier

2024

Gutachter:

* Prof. Dr. Sabine Le Borne
* Prof. Dr. Armin Iske

Datum der mündlichen Prüfung: 15. Juni 2023

doi:10.15480/882.9200


https://orcid.org/0000-0002-3618-2825

Summary

This thesis consists of four parts. The first part starts with a brief introduction when and where partial differential equations (PDEs) arise and a motivation for the numerical solution of PDEs, in particular via the radial basis function - finite difference (RBF-FD) approach. Its main advantages are: it is meshfree and its application is (relatively) straighforward. Furthermore, the structure and the novel contributions of this thesis are stated.

The second part focuses in more detail on the RBF-FD method. RBFs and the RBF-FD approach for the solution of a linear PDE with Dirichlet boundary conditions are introduced. A literature review is conducted to summarize results about the parameters involved in the RBF-FD approach. Through comprehensive numerical experiments, the influence of several of these parameters on the condition numbers of intermediate (local) weight matrices, on the condition number of the resulting (global) stiffness matrix and ultimately on the approximation error of the computed discrete solution to the PDE is studied. Our model problems are three-dimensional PDEs with Dirichlet boundary conditions, namely a Poisson's problem on the cube, a Poisson's problem on the sphere, and a convection-diffusion equation on the cube with a recirculating convective flow. A comprehensive survey on the various components of the basic algorithms for RBF-FD discretization is provided to guide toward a compatible selection of the multitude of RBF-FD parameters in the basic version of RBF-FD such that potential pitfalls such as computationally more expensive setups which not always lead to more accurate numerical solutions are avoided.

The third part concentrates on iterative solvers and preconditioners for the global linear system of equations arising in the application of the RBF-FD approach to the convection-diffusion equation. We compare the solvers generalized minimal residual method and biconjugate gradient stabilized method and discuss left and right preconditioning. Preconditioners based on incomplete LU decompositions and on LU decompositions of hierarchical matrices are studied. The influence of the parameters in the construction of hierarchical matrices is discussed. The advantages and disadvantages of these solvers and preconditioners and the influence of the underlying parameters are studied via comprehensive numerical tests which show computation times, storage costs, numbers of iterations and approximation errors. These insights are utilized for the formulation of general recommendations for the preconditioner setups and the construction of preconditioners for auxiliary stiffness matrices with smaller stencil sizes. Their reliability is illustrated in a proof of concept approach via numerical tests.

The fourth part deals with final conclusions and an outlook to further research directions.

Contents

List of Figures

List of Tables

List of Symbols

Notation	Description	Page List
d	spatial dimension	6
Ω	domain	6
ε	shape parameter, relative (truncation) tolerance	6, 75, 78, 79
$\overline{\Omega}$	closure of Ω	6
n	number (of nodes, RBFs, rows, columns), stencil size, size of I	6, 7, 15, 73
ϕ_ε	generating function with shape parameter ε	6
Φ_{ε,x_i}	RBF with shape parameter ε and center x_i	6
x	evaluation point, solution of linear system	6, 14, 65
x_i	i'th node, point, center, component of x	6, 7, 73
$\mathcal{L}$	(differential) operator, leaves of a tree	7, 76
u	function, (discrete) solution of PDE	7, 12
X_Ω	set of interior nodes	7
$X_{\partial\Omega}$	set of boundary nodes	7
$\partial\Omega$	domain boundary	7
N_I	number of interior nodes	7
N	number of all nodes	7
X	set of all nodes	7
X_j	j'th stencil (or associated set to row index j, see X_i)	7
$\mathcal{I}_j$	index set of nodes included in stencil X_j	7
w_i^j	i'th weight of stencil X_j	7
$\delta_{\ell i}$	Kronecker delta	7
p_u	interpolant of u	7
k_i	i'th basis function, exponent of monomial	7
x_c	evaluation point of the application of the differential operator $\mathcal{L}$, stencil center	7, 9
s_i^j	index of i'th node in stencil X_j	8
w_i	i'th weight	8
Π_ℓ	the space of d-variate polynomials of degree at most ℓ	8
p	polynomial, size of partition P	8, 75
M	dimension of polynomial space Π_ℓ, matrix	8, 73, 75, 79
$\mathcal{R}_j$	function space of RBFs associated with stencil X_j and augmented by polynomials	8
λ_i^j	i'th RBF weight of interpolant s	8

Notation	Description	Page List
$\widetilde{\lambda}_k^j$	k'th polynomial weight of interpolant s	8
p_k	k'th vector in basis of Π_ℓ	8
s	interpolant in $\mathcal{R}_j$, son of v	8, 76
f	function, right hand side function for PDE	8, 12
A_j	RBF matrix w. r. t. stencil X_j	8
P_j	polynomial matrix w. r. t. stencil X_j	8
$\mathbf{0}$	$(M \times M)$ zero matrix	8
0_M	zero vector (of size M)	8
λ^j	RBF weights of interpolant s	8
$\widetilde{\lambda}^j$	polynomial weights of interpolant s	8
f^j	vector of evaluations of f at nodes in stencil X_j	8
$\widetilde{A}_j$	RBF saddle point matrix w. r. t. stencil X_j	8
e_i	i'th unit vector (or relative 2-norm error if $i = 2$, see e_2)	9
$\mathcal{L}\mathbf{\Phi}_j$	RBF part in right hand side of weights computation of stencil X_j	9
$\mathcal{L}\mathbf{p}_j$	polynomial part in right hand side of weights computation of stencil X_j	9
I_n	$(n \times n)$ identity matrix (or n'th subset of I, see I_i)	10
$0_{M\times n}$	$(M \times n)$ zero matrix	10
w^j	weights of stencil X_j	10
$\widetilde{w}^j$	dummy weights of stencil X_j	10
p_k^m	k'th vector in monomial basis of Π_ℓ	11
h_j	inverse of the maximum distance from the center x_j to another stencil node	11
p_k^j	k'th vector in by center x_j shifted monomial basis of Π_ℓ	11
$p_k^{h_j}$	k'th vector in shifted and scaled monomial basis of Π_ℓ	11
X_j^j	by x_j (to origin) shifted j'th stencil	11
$X_j^{h_j}$	by h_j scaled stencil X_j^j	11
g	function for Dirichlet boundary condition	12
u_i	discrete solution of PDE at node x_i	12
$\widetilde{f}_j$	j'th component of $\widetilde{f}$	13
B	stiffness matrix, low-rank factor	13, 75
$\widetilde{f}$	right hand side of linear system arising in RBF-FD	13
$\hat{x}$	solution of perturbed linear system	13
$\hat{A}$	left hand side of perturbed linear system	14
$\hat{b}$	right hand side of perturbed linear system	14
A	left hand side of linear system, low-rank factor	14, 65, 75
b	right hand side of linear system, base, (convective) flow, (index) block	14, 15, 22, 51, 65, 75, 77

Notation	Description	Page List
ϵ_A	relative errors in entries of A	14
ϵ_b	relative errors in entries of b	14
κ	condition number	14
$\mathcal{O}$	Landau symbol	14
h	internodal distance, scaling factor, local distance function	14, 21, 22, 26
$h_{X,\Omega}$	fill distance (w. r. t. nodes X in domain Ω)	15
q_X	separation distance (w. r. t. nodes X)	15
$d_i(n)$	i'th digit (counted from right side) of number n	15
k	parameter to define degree of PHS, number of digits, path length	15, 18, 76
$g_b(n)$	n'th element of van der Corput sequence in base b	15
$p(j)$	j'th prime number	15
n_j	size of stencil X_j	17, 24
Ω_j	domain (w. r. t. stencil X_j)	17
ν	GMQ parameter, diffusion coefficient, number of interior nodes per spatial direction, parameter to define constant spacing function	18, 25, 49, 51
γ	scaling factor for HYB	18
α	scaling factor for HYB and PNP nodes	18, 26
β	scaling factor for HYB	18
ε_j	(optimal) shape parameter w. r. t. stencil X_j	20
λ_i	i'th Lagrange basis function	20
a	scaling order	21
$w^j_{h_j}$	weights of shifted and scaled stencil $X_j^{h_j}$	21
D	order of the (differential) operator $\mathcal{L}$	21, 22
$n_{\min}$	lower bound for n in RBF-FD with polynomial augmentation, parameter in size functions	21, 77
$n_{\min}(\ell)$	degree ℓ dependent lower bound $n_{\min}$	21
Δ	Laplace operator	22
∇	gradient	22
ℓ_j	degree of polynomial augmentation w. r. t. stencil X_j	24
F	three-dimensional version of Franke's function	24
μ	parameter to define a norm, labeling map	26, 76
κ_1	condition number in 1-norm	26
κ_∞	condition number in infinity norm	26
u_s	vector of evaluations of exact PDE solution at interior nodes	27
e_2	relative 2-norm error (or second unit vector, see e_i)	27
c	scaling factor for shape parameter	31
c_1	scaling factor for shape parameter	31
c_2	scaling factor for shape parameter	31

Notation	**Description**	**Page List**
$\widetilde{k}$	parameter to define degree of PHS	64
$\widetilde{\ell}$	degree of polynomial augmentation for PHS	64
y	intermediate solution of right preconditioned linear system	65
TOL	relative residual accuracy	65
$m_{\max}$	maximal number of iterations	65
$x^{(m)}$	m'th iterate	65
P	preconditioner, (block) partition	65, 75, 77
$C(N)$	inverse of the scaling factor for asymptotic complexity	66
I	index set	73
J	index set	73
$M_{i,j}$	entry in i'th row and j'th column of matrix M	73
$\mathcal{F}$	set of full matrices	73
$\mathcal{P}$	power set	75
I_i	i'th subset of I (or $i \times i$ identity matrix, see I_n)	75
J_i	j'th subset of J	75
b_i	i'th (index) block	75
$M\|_b$	submatrix (corresponding to block b)	75
$\mathcal{F}(b)$	set of full matrix blocks (corresponding to block b)	75
r	(local) rank (distribution), root of the tree T	75, 76
$\mathcal{R}(r)$	set of low-rank matrices with rank $\leq r$	75
$\mathcal{R}(r, I \times J)$	set of low-rank matrices w. r. t. $I \times J$ and rank $\leq r$	75
V	vertex set, matrix in SVD	76, 78
v	vertex	76
w	vertex	76
S	son mapping, separator	76, 80
T	(set decomposition) tree (corresponding to I)	76
root	root of a tree	76
level	level-number	76
depth	depth of a tree	76
T'	subtree of T	76
V'	subset of V	76
S'	restriction of S to V'	76
s'	son of v	76
$T(I)$	cluster tree w. r. t. I	76
σ	cluster	76
τ	cluster	76
$T(I \times J)$	block cluster tree w. r. t. $I \times J$	77
$C_{\mathrm{sp,l}}(\tau, P)$	sparsity constant w. r. t. cluster τ and partition P	77
$C_{\mathrm{sp,r}}(\sigma, P)$	sparsity constant w. r. t. cluster σ and partition P	77
$C_{\mathrm{sp}}(P)$	sparsity constant w. r. t. partition P	77
$\mathrm{size}_{T(I)}$	size function w. r. t. $T(I)$	77

Notation	Description	Page List
$\text{size}_{T(I\times J)}$	size function w. r. t. $T(I \times J)$	77
b'	subblock (of the block b)	77
τ'	cluster (subset of τ)	77
σ'	cluster (subset of σ)	77
adm	admissibility condition	77
P^-	near-field	77
P^+	far-field	77
$\mathcal{H}$	hierarchical matrix	77
$\mathcal{H}(r, P)$	set of hierarchical matrices w. r. t. partition P and rank r	77
X_i	associated set to row index i (or i'th stencil, see X_j)	77
Y_j	associated set to column index j	77
X_τ	support of cluster τ	77
Y_σ	support of cluster σ	77
diam	diameter of a cluster	77
dist	distance between clusters	77
η	parameter in η-admissibility condition	78
U	matrix in SVD, factor in LU decomposition	78, 80
Σ	matrix in SVD	78
σ_i	i'th singular value	78
R	best low-rank approximation (with rank $\leq r$)	78
Σ_r	Σ matrix in SVD of R	78
r_2	adaptive rank in Euclidean norm	78
r_F	adaptive rank in Frobenius norm	79
$\mathcal{T}^{\mathcal{R}}_{r\leftarrow s}$	truncation from rank s to rank r	79
$\mathcal{T}^{\mathcal{R}}_{r}$	truncation to rank r	79
$\mathcal{T}^{\mathcal{R}}$	truncation	79
$\mathcal{T}^{\mathcal{R}}_{2,\varepsilon}$	truncation to rank $r_2(\varepsilon)$	79
$\mathcal{T}^{\mathcal{R}}_{F,\varepsilon}$	truncation to rank $r_F(\varepsilon)$	79
P'	partition of coarsened $M' \in \mathcal{H}(r', P')$	79
M'	hierarchical matrix obtained by coarsening hierarchical matrix $M \in \mathcal{H}(r, P)$	79
r'	local rank distribution of coarsened $M' \in \mathcal{H}(r', P')$	79
P_r	row permutation matrix	79
P_c	column permutation matrix	79
M_{D_1}	matrix block of a matrix in nested dissection ordering	80
$M_{D_1\times S}$	matrix block of a matrix in nested dissection ordering	80
M_{D_2}	matrix block of a matrix in nested dissection ordering	80
$M_{D_2\times S}$	matrix block of a matrix in nested dissection ordering	80
$M_{S\times D_1}$	matrix block of a matrix in nested dissection ordering	80
$M_{S\times D_2}$	matrix block of a matrix in nested dissection ordering	80
M_S	matrix block of a matrix in nested dissection ordering	80

Notation	Description	Page List
L_{D_1}	matrix block (corresponding to M_{D_1}) of L factor	80
L_{D_2}	matrix block (corresponding to M_{D_2}) of L factor	80
$L_{S\times D_1}$	matrix block (corresponding to $M_{S\times D_1}$) of L factor	80
$L_{S\times D_2}$	matrix block (corresponding to $M_{S\times D_2}$) of L factor	80
L_S	matrix block (corresponding to M_{D_S}) of L factor	80
U_{D_1}	matrix block (corresponding to M_{D_1}) of U factor	80
$U_{D_1\times S}$	matrix block (corresponding to $M_{D_1\times S}$) of U factor	80
U_{D_2}	matrix block (corresponding to M_{D_2}) of U factor	80
$U_{D_2\times S}$	matrix block (corresponding to $M_{D_2\times S}$) of U factor	80
U_S	matrix block (corresponding to M_{D_S}) of U factor	80
D_1	domain in domain decomposition	80
D_2	domain in domain decomposition	80
L	factor in LU decomposition	80
$S_{\mathcal{H}}(M)$	costs to store an $\mathcal{H}$-matrix $M \in \mathcal{H}(r,P) \subset \mathbb{R}^{I\times I}$	80
$S_{\mathrm{LU}}(M)$	costs to store an $\mathcal{H}$-LU decomposition of $M \in \mathcal{H}(r,P) \subset \mathbb{R}^{I\times I}$	80
$N_{\mathrm{MV}}(M)$	number of arithmetic operations to compute a matrix-vector multiplication for $M \in \mathcal{H}(r,P) \subset \mathbb{R}^{I\times I}$	80
$N_{\mathrm{LU}}(M)$	number of arithmetic operations to apply an $\mathcal{H}$-LU preconditioner of $M \in \mathcal{H}(r,P) \subset \mathbb{R}^{I\times I}$	80
$N_{\mathrm{LUsetup}}(M)$	number of arithmetic operations to setup an $\mathcal{H}$-LU preconditioner of $M \in \mathcal{H}(r,P) \subset \mathbb{R}^{I\times I}$	80
R_b	low-rank approximation of matrix block $M\|_b$	88

List of Abbreviations

BiCGstab biconjugate gradient stabilized
FD finite difference
FE finite element
GA Gaussian
GMQ generalized multiquadric
GMRES generalized minimal residual
HYB hybrid
ILU incomplete LU
IMQ inverse multiquadric
IQ inverse quadratic
LPBiCGstab left preconditioned biconjugate gradient stabilized
LPGMRES left preconditioned generalized minimal residual
MQ multiquadric
NaN not a number
PDE partial differential equation
PHS polyharmonic spline
RBF radial basis function
RBF-FD radial basis function-generated finite difference
RPGMRES right preconditioned generalized minimal residual
SVD singular value decomposition
WLS weighted least squares

1 Introduction

Many processes in various application areas such as fluid mechanics and geosciences, can be described by partial differential equations (PDEs). The solution of a PDE enables the simulation of (and research into) the corresponding process. Nevertheless, these solutions often rely on numerical approximations since their analytical determination can be challenging or even unknown. These numerical approximations are typically based on a discretization of the original infinite-dimensional problem and there are several discretization techniques for the numerical solution of PDEs. In addition to classical finite difference (FD), finite element (FE) and finite volume techniques, a more recent approach employs radial basis functions (RBFs) to generate differentiation stencils on arbitrary point sets. This approach, abbreviated by RBF-FD (radial basis function - finite difference), has gained in popularity since it enjoys several advantages: It is (relatively) straightforward, does not require a mesh and generalizes easily to higher spatial dimensions. However, its application is not quite as blackbox as it may appear at first sight. The computed solution might suffer severely from various sources of errors if RBF-FD parameters are not selected carefully.

Through comprehensive numerical experiments, we study the influence of several of these parameters on the condition numbers of intermediate (local) weight matrices, on the condition number of the resulting (global) stiffness matrix and ultimately on the approximation error of the computed discrete solution to the PDE. The parameters of investigation include the type of RBF (and its shape or other parameters if applicable), the degree of polynomial augmentation, the discretization stencil size, the underlying type of point set (structured/unstructured), and the total number of (interior and boundary) points to discretize the PDE. Our model problem is at first a three-dimensional Poisson's problem on the cube with Dirichlet boundary conditions. Afterwards, numerical tests on a sphere (i. e., changed domain) as well as tests for the convection-diffusion equation (i. e., changed differential operator) are performed to demonstrate that the obtained results generalize to wider problem classes.

The RBF-FD discretization leads to a linear system of equations whose solution is for large problem sizes usually the bottleneck of the whole simulation. Therefore, the study of efficient (iterative) solvers (and preconditioners) for the solution of these linear systems is important. A straighforward solution can be achieved via blackbox solvers such as sparse direct solvers, algebraic multigrid solvers or iterative solvers combined with incomplete LU (ILU) preconditioners, often with no fill-in (ILU(0)). The main drawback of these blackbox solvers is that they often do not rely on specific problem knowledge and can be therefore inferior to problem based (i. e., more finetuned) solvers and pre-

conditioners.

Hierarchical matrices offer one possibility to construct high accuracy (problem dependent) preconditioners and they have not yet been tested for the linear system in the RBF-FD method. Hence, we study hierarchical matrix preconditioners in the context of the RBF-FD approach and discuss what could and what should not be adopted from the literature about hierarchical matrix preconditioners for other sparse matrices.

The remainder of this thesis is organized as follows. The RBF related content is in Chapter 2 (which is an extended version of [74]), namely

- definition of RBFs,
- introduction to RBF interpolation,
- derivation of the RBF-FD method to solve a linear PDE with Dirichlet boundary conditions,
- discussion of further RBF-FD setup details such as the node generation, the stencil formation and the computation of the stencil weights,
- numerical tests of the approximation errors in different RBF-FD setups,
- conclusions and statement of the recommended RBF-FD setup.

This provides a comprehensive survey on the various components of the basic algorithms for RBF-FD discretization to steer away from potential pitfalls such as computationally more expensive setups which not always lead to more accurate numerical solutions. We guide toward a compatible selection of the multitude of RBF-FD parameters in the basic version of RBF-FD. For many of its components we refer to the literature for more advanced versions.

We concentrate in Chapter 3 on the iterative solution (i. e., mainly the construction of suitable preconditioners) for the linear system in the basic version of the RBF-FD approach, namely

- choice of the solver,
- ILU(0) preconditioners and their advantages and disadvantages,
- introduction to hierarchical matrices,
- review of hierarchical matrix solvers and preconditioners for other sparse matrices,
- setup of hierarchical matrix preconditioners for RBF-FD sparse matrices (and their implementation in the H2Lib),
- numerical tests of various solver and preconditioner setups,
- numerical tests with smaller stencil preconditioners (which are an extended and modified version of [75]),
- conclusions and statement of the recommended solver (and preconditioner) setup.

The novel contributions of this thesis (and of the corresponding papers [74, 75]) consist of

- a comprehensive view and comparison of basic RBF-FD methods from the literature;
- general recommendations for parameter setups in basic versions of RBF-FD discretizations (e. g., a new scaling law for the stencil size and the degree of polynomial augmentation on Cartesian node distributions and a general recommendation to shift stencils to the origin if polynomials of large degrees are augmented);
- numerical tests and recommendations on the development of hierarchical matrix preconditioners to RBF-FD sparse matrices;
- the introduction of preconditioners that are based on an auxiliary RBF-FD stiffness matrix with a smaller stencil size.

2 Radial basis function - finite difference (RBF-FD) method

2.1 Introduction

(Linear) partial differential equations can be solved numerically by a suitable discretization followed by the solution of a linear system of equations. Among possible discretization techniques, the radial basis function - finite difference (RBF-FD) method can be viewed as a generalization of the finite difference (FD) method to unstructured point sets. The earlier papers on RBF-FD include [119, 120]. Since then, it has gained in popularity and entered into several applications including geosciences [42, 43], heat flow [80], the financial sector [83], linear elasticity [114, 111], fluid mechanics [20, 64], magnetohydrostatics [81] and neuronal dynamics [93].

As in classical finite difference methods, RBF-FD replaces (linear) differential operators by differentiation stencils which encode formulas for weighted sums of function values at neighboring points. The weights are chosen such that the formulas become exact for a set of test functions which typically include radial as well as polynomial functions.

The discretization error in the numerical solution of a boundary value problem is affected by several aspects of the solution process including the underlying point set (the number of interior/boundary nodes and their distribution), the number (and distribution) of nodes in the discretization stencil as well as the RBF (including its shape or other parameters) and the degree of polynomial augmentation used to generate the stencil weights.

The application of RBF-FD discretization can lead to (auxiliary) highly ill-conditioned linear systems of equations to be solved for the weights [15], resulting in instability which prevents convergence of the numerical to the exact solution [42, 40, 68]. One possibility to address the ill-conditioning lies in a scaling of the shape parameter which, however, leads to a stagnation effect [5, 111]. Another possibility lies in the use of hybrid kernels [87, 85], another in the transition to stable versions of RBFs, e. g. RBF-GA [44, 76, 71], RBF-RA [48, 129] or RBF-QR [68]. These, however, typically incur increased computational cost. A computationally even more expensive possibility to address the ill-conditioning is provided by (sufficiently high) extended precision [41]. Further possibilities to address the ill-conditioning are spatially variable shape parameters [46, 16, 98], Tikhonov regularization [117] or smoothing splines [36, 136]. More recent papers often suggest to use polyharmonic splines (PHS) in combination with high order polynomials

to generate the stencil weights. In [41, 42], the Gaussian (GA) and PHS generating functions are studied and compared w. r. t. polynomial augmentation, extended precision and stable algorithms. In [102], PHS with polynomial augmentation (for a fixed stencil size and a numerically determined optimal degree of polynomial augmentation) is compared to the stable RBF-GA method with a small shape parameter (i. e., representing the limit case $\varepsilon \to 0$). The advantages and disadvantages of the choice of parameters and the performance of the PHS generating function with polynomial augmentation are further studied in [7, 6, 10, 38, 107, 61].

In this chapter, we will review some of these approaches with a focus on the most commonly used and (to us) most promising ones. We will discuss the multitude of involved parameter choices, illustrate their influence through numerical tests, and draw comparisons between the different approaches. There already exist quite a few numerical studies for the performance of RBF-FD approximation and the solution of partial differential equations, but we address yet another angle that we have not found in the literature yet. Namely, the novel contributions of this chapter consist of

- a comprehensive view and comparison of basic but often still competitive RBF-FD methods from the literature, in particular comprehensive numerical tests to illustrate the condition numbers of weight and stiffness matrices side-by-side with the discretization errors for a broad set of parameter choices across the landscape of RBF-FD discretization; the intention is to illustrate the ill-conditioning of weight matrices as a cause for numerical instabilities and the scaling of the shape parameter which is introduced to control this ill-conditioning as a cause for stagnation, show both the potential and limitations of individual RBF classes and offer a comparison to alternative RBF-FD setups (see Subsection 2.4.2 to Subsection 2.4.4);
- a new scaling law for the stencil size and the degree of polynomial augmentation for hybrid (HYB) [85] and PHS generating functions on Cartesian node distributions (see Subsections 2.4.3 and 2.4.4);
- a general recommendation to shift stencils to the origin if polynomials of large degrees are augmented (see Subsections 2.4.3 and 2.4.4);
- a detailed (numerical) comparison of the HYB and the PHS generating functions for the case of polynomial augmentation (see Subsection 2.4.4);
- general recommendations for parameter setups in the basic version of RBF-FD discretizations (see Subsection 2.4.7).

While in this chapter all linear systems are solved directly, our focus on condition numbers of the resulting stiffness matrices will also be of interest in Chapter 3 for the subsequent development of iterative solvers and preconditioners for the systems.

The remainder of this chapter is organized as follows. In section 2.2, we review RBFs, RBF-FD approximation of linear differential operators and its application to the discretization of linear partial differential equations. The details pertaining to the node generation, the stencil formation and computation of the stencil weights, the choice of the RBF and the role of polynomial augmentation are discussed in section 2.3. Section

2.4 is devoted to comprehensive numerical tests for a multitude of parameter choices in RBF-FD discretization first applied to the three-dimensional Poisson's equation with Dirichlet boundary conditions. In order to illustrate that qualitative results also apply to a wider class of model problems, we provide afterwards additional numerical results on a different domain, i. e., a sphere, as well as for a different partial differential equation, i. e., the convection-diffusion equation with a recirculating convection. A conclusion and an outlook are presented in Section 2.5.

2.2 The RBF-FD method

In this section, we introduce radial basis functions as a tool for the approximation of linear differential operators (Subsection 2.2.1), explain the discretization of a partial differential equation with Dirichlet boundary conditions by means of the RBF-FD method (Subsection 2.2.2) and summarize the various types of errors that may and typically do occur in the overall numerical RBF-FD solution process (Subsection 2.2.3).

2.2.1 RBFs and RBF-FD approximation

Radial basis functions typically refer to a set of functions that are characterized by individual centers but are otherwise identical and radial in shape, i. e., shifted copies of one another, the function value at a point depending only on its distance to the center. A motivation for the definition of RBFs, or more general kernel functions [34, 33, 35], and their application in multivariate interpolation (or approximation) problems is the Mairhuber-Curtis theorem [127]. The message of this theorem is that interpolation problems in $d > 1$ dimensions require structured nodes or node dependent basis functions to enable that their solution is for all node sets given by a regular linear system. Formally, RBFs can be defined as follows.

Definition 2.1. *For a spatial dimension $d \in \mathbb{N}$, an open domain $\Omega \subseteq \mathbb{R}^d$, a shape parameter $\varepsilon > 0$, a set of pairwise distinct nodes $\{x_1, \ldots, x_n\} \subset \overline{\Omega}$ with $n \in \mathbb{N}$, and a generating function $\phi_\varepsilon \colon [0, \infty) \to \mathbb{R}$, we define radial basis functions (RBFs) centered at the nodes x_i, $i \in \{1, \ldots, n\}$, as*

$$\Phi_{\varepsilon,x_i} \colon \overline{\Omega} \to \mathbb{R}, \quad \Phi_{\varepsilon,x_i}(x) := \phi_\varepsilon \left(\|x - x_i\|_2 \right), \quad i \in \{1, \ldots, n\}.$$

Sometimes, the generating function ϕ_ε is referred to as a kernel function or even as a radial basis function itself. Ideally, the functions Φ_{ε,x_i}, $i \in \{1, \ldots, n\}$, are linearly independent and hence form a basis of their span so that the terminology *basis function* is justified. Furthermore, in order to later be able to guarantee the existence of unique solutions of (discrete) interpolation problems using radial basis functions, we will require the kernel function to be (conditionally) positive definite, see Definition 2.2. Typical examples for such kernel functions are listed in Table 2.1.

We list several reasons for the popularity of RBFs for multivariate interpolation and approximation [126]:

- RBFs can be used in any spatial dimension.
- RBFs are applicable to arbitrarily scattered data. In particular, no mesh is required.
- Interpolants/approximants have a simple representation and inherit the smoothness of the radial basis functions.

RBFs can be used to approximate the action of a linear differential operator ([111, Proposition 2.2.7 and Algorithm 5.1] explain how to deal with differential operators that are a linear combination of differential operators) as follows. Let $\mathcal{L}$ denote a linear differential operator, $u\colon \overline{\Omega} \to \mathbb{R}$ be a (in Ω) sufficiently smooth function, and let

$$X_\Omega := \{x_1, \dots, x_{N_I}\} \subset \Omega, \qquad X_{\partial\Omega} := \{x_{N_I+1}, \dots, x_N\} \subset \partial\Omega \tag{2.1}$$

be sets of distinct interior and boundary nodes, resp., with $N_I, N \in \mathbb{N}$, $N_I \leq N$ (see Subsection 2.3.1 for more details about possibilities for the generation of these node sets). Such node sets $X := X_\Omega \cup X_{\partial\Omega}$ also occur in finite difference discretizations, but now the nodes may be scattered and need not be connected by edges of a (structured tensor product) grid.

For a node $x_j \in X_\Omega$, $j \in \{1, \dots, N_I\}$, and stencil size $n \leq N$, let $X_j \subseteq X$ be a stencil associated to the (stencil) center x_j (see Subsection 2.3.2 for more information on the stencil computation). We use $\mathcal{I}_j \subseteq \{1, \dots, N\}$ to denote the index set of nodes included in the stencil X_j, i. e.,

$$X_j = \{x_{s_1^j}, \dots, x_{s_n^j}\} = \{x_i \in X : i \in \mathcal{I}_j\} \quad \text{for } \mathcal{I}_j := \{s_1^j, \dots, s_n^j\}. \tag{2.2}$$

The goal is to determine weights $\{w_1^j, \dots, w_n^j\} \subset \mathbb{R}$ such that the differential operator applied to the function u and evaluated at x_j can be expressed (approximately) as a weighted sum of function values at the nodes in the stencil X_j, i. e.,

$$\mathcal{L}u(x_j) := \mathcal{L}u(x)|_{x=x_j} \approx \sum_{i=1}^{n} w_i^j u(x_{s_i^j}). \tag{2.3}$$

The stencil weights are determined such that the weighted sum yields the correct result for the interpolant w. r. t. the stencil nodes, i. e., equality is enforced for (basis) functions $k_1, \dots, k_n\colon \Omega \to \mathbb{R}$. If these are cardinal basis functions w. r. t. the stencil nodes in X_j, i. e., if there holds $k_\ell(x_{s_i^j}) = \delta_{\ell i}$ with the Kronecker delta, then

$$p_u\colon \Omega \to \mathbb{R}, \quad p_u(x) = \sum_{i=1}^{n} u(x_{s_i^j}) k_i(x)$$

is the interpolant of u, and the application of a differential operator $\mathcal{L}$ at some $x_c \in \Omega$

is given by

$$\mathcal{L}p_u(x_c) = \sum_{i=1}^{n} u(x_{s_i^j})\mathcal{L}k_i(x_c) =: \sum_{i=1}^{n} u(x_{s_i^j})w_i \tag{2.4}$$

for weights defined by $w_i := \mathcal{L}k_i(x_c)$, $i \in \{1\ldots,n\}$.

We will next illustrate the computation of these weights for typical RBF spaces with (or without) polynomial augmentation. Let $\{\Phi_{\varepsilon,x_i} \mid i \in \mathcal{I}_j\}$ denote the n radial basis functions associated with the nodes of the stencil X_j, and let

$$\Pi_\ell := \operatorname{span}\left\{p : \Omega \to \mathbb{R},\ p(x) = \prod_{j=1}^{d} x_j^{k_j} \mid k_j \in \mathbb{N}_0, \sum_{j=1}^{d} k_j \le \ell\right\} \tag{2.5}$$

denote the space of d-variate polynomials of degree at most $\ell \in \mathbb{N}_0$. There holds

$$\dim \Pi_\ell = \binom{\ell+d}{d} =: M, \tag{2.6}$$

and we denote a basis of Π_ℓ by $\{p_1,\ldots,p_M\}$. The n-dimensional constrained function space of RBFs associated with the stencil X_j (2.2) and augmented by polynomials up to degree ℓ is defined by

$$\begin{aligned}\mathcal{R}_j := \Bigg\{ s : \Omega \to \mathbb{R},\ s(x) = \sum_{i=1}^{n} \lambda_i^j \Phi_{\varepsilon,x_{s_i^j}}(x) + \sum_{k=1}^{M} \widetilde{\lambda}_k^j p_k(x) \mid \lambda_i^j, \widetilde{\lambda}_k^j \in \mathbb{R} \\ \text{such that } \sum_{i=1}^{n} \lambda_i^j p_k(x_{s_i^j}) = 0 \text{ for all } k \in \{1,\ldots,M\}\Bigg\}.\end{aligned} \tag{2.7}$$

It is straightforward that (the weights of) an interpolant $s \in \mathcal{R}_j$ of a function $f\colon \Omega \to \mathbb{R}$ can be computed as the solution of the linear system

$$\begin{bmatrix} A_j & P_j \\ P_j^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \lambda^j \\ \widetilde{\lambda}^j \end{bmatrix} = \begin{bmatrix} f^j \\ 0_M \end{bmatrix} \tag{2.8}$$

with coefficient and right hand side vectors

$$\lambda^j := \begin{bmatrix} \lambda_1^j \\ \vdots \\ \lambda_n^j \end{bmatrix}, \quad \widetilde{\lambda}^j := \begin{bmatrix} \widetilde{\lambda}_1^j \\ \vdots \\ \widetilde{\lambda}_M^j \end{bmatrix}, \quad f^j := \begin{bmatrix} f\left(x_{s_1^j}\right) \\ \vdots \\ f\left(x_{s_n^j}\right) \end{bmatrix}, \quad 0_M := \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^M$$

and system matrix

$$\widetilde{A}_j := \begin{bmatrix} A_j & P_j \\ P_j^T & \mathbf{0} \end{bmatrix} := \begin{bmatrix} \Phi_{\varepsilon,x_{s_1^j}}(x_{s_1^j}) & \cdots & \Phi_{\varepsilon,x_{s_n^j}}(x_{s_1^j}) & p_1(x_{s_1^j}) & \cdots & p_M(x_{s_1^j}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \Phi_{\varepsilon,x_{s_1^j}}(x_{s_n^j}) & \cdots & \Phi_{\varepsilon,x_{s_n^j}}(x_{s_n^j}) & p_1(x_{s_n^j}) & \cdots & p_M(x_{s_n^j}) \\ p_1(x_{s_1^j}) & \cdots & p_1(x_{s_n^j}) & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_M(x_{s_1^j}) & \cdots & p_M(x_{s_n^j}) & 0 & \cdots & 0 \end{bmatrix}. \tag{2.9}$$

The interpolation problem has a unique solution if this matrix $\widetilde{A}_j$ is regular. This can be guaranteed if the kernel function and node set X satisfy the properties given in the Definitions 2.2 and 2.3, resp.

Definition 2.2. *The generating function $\phi : [0, \infty) \to \mathbb{R}$ is called conditionally positive definite of order $\ell \in \mathbb{N}$ if*

$$\sum_{j=1}^{N} \sum_{k=1}^{N} \lambda_j \lambda_k \phi(\|x_j - x_k\|_2) > 0 \tag{2.10}$$

for any (finite) set $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^n$ of distinct nodes and coefficients $\lambda_j, j = 1, \dots, N$, $\lambda_j \neq 0$ for at least one $j \in \{1, \dots, N\}$, satisfying

$$\sum_{j=1}^{N} \lambda_j p(x_j) = 0$$

for all polynomials $p \in \Pi_{\ell-1}$.
The generating function ϕ is called positive definite if (2.10) holds without any restrictions on λ_j imposed through polynomials.

Definition 2.3. *A (finite) set $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^n$ of distinct nodes is called Π_ℓ-unisolvent if there is no nonzero polynomial in Π_ℓ that vanishes on all nodes in X, i. e., if*

$$p(x_j) = 0 \quad \forall j \in \{1, \dots, N\} \quad \Longrightarrow \quad p = 0.$$

More details concerning (conditionally) positive definite functions are included in most textbooks on radial basis functions, see e. g. [127, 17]. In terms of matrices, Π_ℓ-unisolvency guarantees that the submatrix P_j in (2.9) has full column rank, and if in addition the generating function is conditionally positive definite (of order $\ell + 1$ or less), then $\widetilde{A}_j$ is regular.

The cardinal functions of the space $\mathcal{R}_j$ may now be represented by

$$k_i(x) = \begin{bmatrix} \Phi_{\varepsilon, x_{s_1^j}}(x) & \cdots & \Phi_{\varepsilon, x_{s_n^j}}(x) & p_1(x) & \cdots & p_M(x) \end{bmatrix} \widetilde{A}_j^{-1} \begin{bmatrix} e_i \\ 0_M \end{bmatrix}$$

where $e_i \in \mathbb{R}^n$ denotes the i'th unit vector. Application of the differential operator and evaluation at the stencil center $x_c := x_j$ yields

$$\begin{aligned} w_i^j &\overset{(2.4)}{=} \mathcal{L}k_i(x_j) \\ &= \big[\underbrace{\mathcal{L}\Phi_{\varepsilon, x_{s_1^j}}(x_j) \ \cdots \ \mathcal{L}\Phi_{\varepsilon, x_{s_n^j}}(x_j)}_{=:(\mathcal{L}\boldsymbol{\Phi}_j)^T} \ \underbrace{\mathcal{L}p_1(x_j) \ \cdots \ \mathcal{L}p_M(x_j)}_{=:(\mathcal{L}\mathbf{p}_j)^T} \big] \widetilde{A}_j^{-1} \begin{bmatrix} e_i \\ 0_M \end{bmatrix} \end{aligned}$$

for a single weight and

$$\begin{bmatrix} w_1^j & \dots & w_n^j \end{bmatrix} = \begin{bmatrix} (\mathcal{L}\mathbf{\Phi}_j)^T & (\mathcal{L}\mathbf{p}_j)^T \end{bmatrix} \widetilde{A}_j^{-1} \begin{bmatrix} I_n \\ 0_{M\times n} \end{bmatrix}$$

for all weights with identity matrix $I_n \in \mathbb{R}^{n\times n}$ and zero matrix $0_{M\times n} \in \mathbb{R}^{M\times n}$. Augmentation of the matrix on the right to an $(n+M)\times(n+M)$ identity matrix and introduction of dummy variables $\widetilde{w}_i^j$ (which may later be discarded since they originated from the arbitrary augmentation to the identity matrix) leads to the linear system of equations

$$\underbrace{\begin{bmatrix} A_j & P_j \\ P_j^T & \mathbf{0} \end{bmatrix}}_{=\widetilde{A}_j (2.9)} \begin{bmatrix} w^j \\ \widetilde{w}^j \end{bmatrix} = \begin{bmatrix} \mathcal{L}\mathbf{\Phi}_j \\ \mathcal{L}\mathbf{p}_j \end{bmatrix} \quad \text{with} \quad w^j := \begin{bmatrix} w_1^j \\ \vdots \\ w_n^j \end{bmatrix}, \quad \widetilde{w}^j := \begin{bmatrix} \widetilde{w}_1^j \\ \vdots \\ \widetilde{w}_M^j \end{bmatrix} \tag{2.11}$$

for the desired stencil weights w_i^j in the approximation (2.3) [42]. The variables $\widetilde{w}^j$ are sometimes called dummy weights or dummy values since they are not used in the Approximation (2.3), or Lagrange multipliers since the stencil weights w^j are the solution of the convex minimization problem

$$\min_{w^j} \ \frac{1}{2} \left(w^j\right)^T A_j w^j - \left(w^j\right)^T \mathcal{L}\mathbf{\Phi}_j \quad \text{subject to} \quad P_j^T w^j = \mathcal{L}\mathbf{p}_j$$

and $\widetilde{w}^j$ are the corresponding Lagrange multipliers [41]. In the literature, the Approximation (2.3) with weights (2.11) is usually called RBF-FD discretization or RBF-FD approximation, sometimes also *RBF-FD interpolation* [41, 137] to indicate that the differential operator $\mathcal{L}$ in Approximation (2.3) may be the identity operator (i. e., the zeroth derivative is approximated). Furthermore, the matrices $\widetilde{A}_j$ in the stencil weights computations (2.11) are called RBF-FD interpolation matrices in [107].

The following definitions can be used to describe when the matrices $\widetilde{A}_j$ and A_j are regular.

Definition 2.4. *An infinitely smooth function $f\colon (0,\infty) \to \mathbb{R}$ is completely monotonic if and only if*

$$(-1)^k \frac{d^k f(r)}{dr^k} \geq 0$$

for all $k \in \mathbb{N}_0$ and $r > 0$. An RBF Φ_{ε,x_i} is (for each center x_i) strictly conditionally positive definite of order $\ell \in \mathbb{N}$ if and only if, its generating function ϕ_ε is conditionally positive of order ℓ.

Without polynomial augmentation, the saddle point system (2.11) simplifies to $A_j w^j = \mathcal{L}\mathbf{\Phi}_j$ and A_j is regular, if (at least) one of the following conditions is fulfilled

- $\phi_\varepsilon(\sqrt{r})$ restricted to $(0,\infty)$ is completely monotonic and nonconstant,

- $\phi_\varepsilon(0) \geq 0$ and the first derivative of $\phi_\varepsilon(\sqrt{r})$ restricted to $(0, \infty)$ is completely monotonic and nonconstant [17].

However, there exist several reasons to add polynomials in RBF-FD approximation [41, 42], including the following.

- For certain RBFs, the matrix A_j in (2.11) can be singular (e. g., if $\Phi_\varepsilon \colon \Omega \to \mathbb{R}$, $\Phi_\varepsilon(x) := \phi_\varepsilon(\|x\|_2)$ is strictly conditionally positive of order k) whereas the matrix $\widetilde{A}_j$ is regular for an appropriately chosen polynomial degree ℓ (i. e., with $\ell \geq k-1$ if the stencil X_j is unisolvent for Π_ℓ) [17]. The Linear System of Equations (2.11) is uniquely solvable if $P_j^T w^j = \mathcal{L}\mathbf{p}_j$ is solvable [21].
- Exact/accurate representation of low order polynomials (in particular constants) instead of an oscillatory representation which is often obtained without polynomial augmentation.
- The accuracy of the approximation in (2.3) may be significantly improved.
- Stagnation errors may be reduced.

In the following, we discuss the choice of the polynomial basis $\{p_1, \ldots, p_M\}$ of Π_ℓ (2.5), modifications to a stencil X_j (2.2) and their benefits (see [57, 26, 111, 112] for more details).

Definition 2.5. *We denote the monomial basis of Π_ℓ by $\{p_1^m, \ldots, p_M^m\}$. For a stencil X_j with center x_j, we define*

1. *$h_j := 1/\max\{\|x_j - x\|_2 : x \in X_j\}$ as the inverse of the maximum distance from the center x_j to another stencil node;*
2. *$\{p_1^j, \ldots, p_M^j\} := \{p_1^m(\cdot - x_j), \ldots, p_M^m(\cdot - x_j)\}$ as the monomial basis of Π_ℓ shifted by the center x_j;*
3. *$\{p_1^{h_j}, \ldots, p_M^{h_j}\} := \{p_1^m(h_j(\cdot - x_j)), \ldots, p_M^m(h_j(\cdot - x_j))\}$ as the basis $\{p_1^j, \ldots, p_M^j\}$ of Π_ℓ scaled by h_j;*
4. *$X_j^j := \{x - x_j : x \in X_j\}$ as the stencil X_j shifted by the center x_j (to the origin);*
5. *$X_j^{h_j} := \{h_j(x - x_j) : x \in X_j\}$ as the stencil X_j^j scaled by h_j.*

Remarks 2.6 and 2.7 illustrate that using a shifted stencil X_j^j is equivalent to using shifted bases of the RBF and the polynomial spaces (i. e., a modified function space $\mathcal{R}_j$ (2.7)).

Remark 2.6. *Using a monomial basis shifted by the center x_j (of stencil X_j) such as $\{p_1, \ldots, p_M\} = \{p_1^j, \ldots, p_M^j\}$ (or $\{p_1, \ldots, p_M\} = \{p_1^{h_j}, \ldots, p_M^{h_j}\}$) in (2.11) is equivalent to using a shifted stencil such as X_j^j (or $X_j^{h_j}$) with the monomial basis $\{p_1, \ldots, p_M\} = \{p_1^m, \ldots, p_M^m\}$.*

Remark 2.7. *If the stencil X_j is replaced by the shifted stencil X_j^j (and the shifted evaluation point is used, i. e., the origin instead of x_j), then the matrix A_j and the*

vector $\mathcal{L}\mathbf{\Phi}_j$ *in* (2.11) *are unchanged since*

$$\Phi_{\varepsilon,x_{s_i^j}}(x_{s_k^j}) = \phi_\varepsilon\left(\|x_{s_k^j} - x_{s_i^j}\|_2\right) = \phi_\varepsilon\left(\|x_{s_k^j} - x_j - (x_{s_i^j} - x_j)\|_2\right) = \Phi_{\varepsilon,x_{s_i^j}-x_j}(x_{s_k^j} - x_j)$$

holds for all $i, k \in \{1, \dots, n\}$. *Hence, the weights* w^j *are unchanged.*

An example (based on [38]) to illustrate a benefit (i. e., avoiding unnecessary numerical instabilities by doing analytical computations instead [111]) of shifting a stencil X_j (or of changing the polynomial basis $\{p_1, \dots, p_M\}$) is shown in Remark 2.8.

Remark 2.8. *The entries of* $\mathcal{L}\mathbf{p}_j$ *(and* P_j*) in* (2.11) *cannot be computed a priori if the monomial basis* $\{p_1, \dots, p_M\} = \{p_1^m, \dots, p_M^m\}$ *of* Π_ℓ *is utilized. However, the evaluation of* $\mathcal{L}\mathbf{p}_j$ *greatly simplifies in the case of a monomial basis shifted by the center* x_j *(of stencil* X_j*) such as* $\{p_1, \dots, p_M\} = \{p_1^j, \dots, p_M^j\}$ *(or* $\{p_1, \dots, p_M\} = \{p_1^{h_j}, \dots, p_M^{h_j}\}$*): For the Laplace operator in* $d = 2$ *dimensions, the entries corresponding to the shifted monomials* $(x - x_j)^2$ *and* $(y - x_j)^2$ *(or* $h_j(x - x_j)^2$ *and* $h_j(y - x_j)^2$*) are* 2*, all other entries of* $\mathcal{L}\mathbf{p}_j$ *are zero. Furthermore (in the usual cases with* $x_j \in X_j$*), the row of* P_j *corresponding to the evaluation at the center* x_j *is a unit vector.*

Further reasons to use shifted and scaled stencils (such as working with local dimensionless coordinates) are given in [111].

2.2.2 RBF-FD for partial differential equations

In Subsection 2.2.1, we derived an approximation of the application of a linear differential operator to a function and its evaluation at a fixed node by a sum of weighted function values at stencil nodes. We next extend this approach to the discretization of a linear partial differential equation

$$\mathcal{L}u(x) = f(x) \quad \forall x \in \Omega, \tag{2.12a}$$
$$u(x) = g(x) \quad \forall x \in \partial\Omega, \tag{2.12b}$$

where $f\colon \Omega \to \mathbb{R}$ is a sufficiently smooth function and $g\colon \partial\Omega \to \mathbb{R}$ is a function to specify the Dirichlet boundary conditions.

For each interior node $x_j \in X_\Omega,\ j \in \{1, \dots, N_I\}$, of the point set $X = X_\Omega \cup X_{\partial\Omega} \subset \overline{\Omega}$, we compute a stencil $X_j \subset X$ of size n (2.2) and stencil weights $\{w_1^j, \dots, w_n^j\}$ by solving (2.11). The stencil weights are used to compute approximations u_i to the solution $u(x_i)$ of (2.12a), (2.12b) at all interior nodes $x_i \in X_\Omega$, i. e., $u_i \approx u(x_i)$ for all $x_i \in X_\Omega, i \in \{1, \dots, N_I\}$ by enforcing equality in (2.3), i. e.,

$$\sum_{i=1}^{n} w_i^j u_{s_i^j} = f(x_j), \quad j \in \{1, \dots, N_I\}. \tag{2.13}$$

Given the Dirichlet boundary data from (2.12b), we set $u_{s_i^j} := g(x_{s_i^j})$ for all $x_{s_i^j} \in \partial\Omega$, substitute this boundary information into (2.13) and obtain

$$\sum_{\substack{i\in\{1,\dots,n\}\\ \text{s.t. } x_{s_i^j}\in\Omega}} w_i^j u_{s_i^j} = \underbrace{f(x_j) - \sum_{\substack{i\in\{1,\dots,n\}\\ \text{s.t. } x_{s_i^j}\in\partial\Omega}} w_i^j g(x_{s_i^j})}_{=:\widetilde{f}_j}, \qquad j \in \{1,\dots,N_I\}.$$

Hence, we define the right hand side vector $\widetilde{f} := (\widetilde{f}_1,\dots,\widetilde{f}_{N_I})^T \in \mathbb{R}^{N_I}$ and the global stiffness matrix $B \in \mathbb{R}^{N_I\times N_I}$, containing row-wise the weights of the interior nodes of the j'th stencil,

$$B_{j,\ell} := \begin{cases} w_i^j & : \exists i \in \{1,\dots,n\} \text{ such that } \ell = s_i^j \in \mathcal{I}_j, \\ & \quad \text{i. e., the interior node } x_\ell = x_{s_i^j} \text{ is in the stencil } X_j, \\ 0 & : \text{else,} \end{cases} \tag{2.14}$$

for all $j,\ell \in \{1,\dots,N_I\}$. The solution $u \in \mathbb{R}^{N_I}$ of the linear system of equations

$$Bu = \widetilde{f} \tag{2.15}$$

yields the RBF-FD approximation $u_j \approx u(x_j)$ to the solution of the Partial Differential Equation (2.12a), (2.12b) at the interior nodes x_j for $j \in \{1,\dots,N_I\}$.

2.2.3 Sources for errors in the RBF-FD method

There are several sources that contribute to (numerical) approximation errors $|u_j - u(x_j)|$, $j \in \{1,\dots,N_I\}$ in the RBF-FD method where $u \in \mathbb{R}^{N_I}$ denotes the computed solution of (2.15). First of all, there occurs a discretization error when approximating the differential operator by a weighted sum of function values at the stencil nodes (2.3). The computation of the weights (2.11) requires the solution of (small, possibly ill-conditioned) linear systems of equations subject to floating point arithmetic [56]. Remark 2.8 illustrates that the amount of floating point computations (and the associated inaccuracies) to evaluate $\mathcal{L}\mathbf{p}_j$ (and P_j) in (2.11) can be reduced (i. e., they are computed analytically instead) by choosing a different basis $\{p_1,\dots,p_M\}$ of Π_ℓ (2.5) for each stencil X_j. The subsequent computation of the RBF-FD approximate solution u requires the solution of another (now large and sparse) linear system of equations whose matrix coefficients consist of those weights, i. e., solutions of ill-conditioned systems. Errors in u can hence be attributed to errors in the stiffness matrix B as well as errors in the right hand side $\widetilde{f}$ in (2.15) which might result in amplified relative errors in u if the stiffness matrix B is ill-conditioned.

To illustrate the propagation of errors, we recall the well-known result from numerical linear algebra bounding the relative error between the solution $\hat{x}$ of a perturbed linear

system of equations $\hat{A}\hat{x} = \hat{b}$ and the solution x of $Ax = b$ w. r. t. the relative errors in the matrix entries, ϵ_A, and right hand side, ϵ_b,

$$\frac{\|\widehat{x} - x\|}{\|x\|} \leq \frac{\mathcal{K}(A)}{1 - \epsilon_A \cdot \mathcal{K}(A)}(\epsilon_A + \epsilon_b), \tag{2.16}$$

if $\epsilon_A \cdot \kappa(A) < 1$. If we assume for the weight matrices and right hand sides in (2.11) relative errors of order $\epsilon_{A_j}, \epsilon_{b_j} \in \mathcal{O}(10^{-16})$ (i. e., rounding errors when computing in double precision accuracy), then the relative error in the computed weights is bounded by $\kappa(A_j) \cdot 10^{-16}$. Hence, we might lose up to $\lfloor \log_{10} \kappa(A_j) \rfloor$ digits of accuracy in the weights which in turn define via (2.14) the matrix entries of the stiffness matrix B in (2.15), leading to a possibly large relative error ϵ_B in these matrix entries. Hence, both the condition numbers of the weight matrices $\widetilde{A}_j$ in (2.11) and the stiffness matrix B in (2.15) give insight into the numerical approximation errors observed in the RBF-FD solution of a partial differential equation. In [41], it is stated that the smallest achievable error (i. e., the accuracy barrier) in double precision w. r. t. floating point arithmetic is given by

$$10^{-15}/h^2 \quad \text{(with } h \text{ denoting the internodal distance).} \tag{2.17}$$

2.3 Node and stencil generation, choice of RBFs and polynomial augmentation

In this section, we discuss some methods for the node generation (Subsection 2.3.1), the stencil formation (Subsection 2.3.2), the choice of the RBF (Subsection 2.3.3) and the role of polynomial augmentation in RBF-FD (Subsection 2.3.4).

2.3.1 Generation of node distributions

The underlying application may already provide the set of (scattered) nodes. However, we may have the luxury to generate our own node set for a given domain and its boundary which is typically the case for the numerical solution of partial differential equations. We discuss several options to do so and refer to the literature for further sophisticated methods that may be necessary for more complex applications. However, nodes obtained via a mesh generator (by simply ignoring the connectivity information) are not needed [24] and can even decrease the numerical stability [108].

The different ways to generate a node distribution can be roughly classified as regular/structured (e. g., Cartesian, Chebyshev or hexagonal) [85] or irregular/unstructured/scattered (e. g., coordinates generated by some type of random number generator). There are also variants that are neither structured nor (completely) unstructured such as node sets generated by a sequence of quasi-random numbers, or initially regular node distributions that have been subsequently perturbed [97]. Sophisticated algorithms to

create node distributions in parallel, with variable node density or strategies for local modification and refinement can be found in [27, 113, 42, 91, 84, 30, 108, 101, 111, 32].

Two important quantities which are relevant for the RBF interpolation and approximation errors are the fill distance and separation distance of a point set.

Definition 2.9. *Given a node set X in a domain Ω, the fill distance $h_{X,\Omega}$ (radius of largest circle not intersecting with X) and the separation distance q_X (shortest distance of two disjoint nodes) are defined by*

$$h_{X,\Omega} := \sup_{x\in\Omega} \min_{x_j\in X} \|x - x_j\|_2 \quad \text{and} \quad q_X := \frac{1}{2} \min_{\substack{x_i,x_j\in X \\ j\neq i}} \|x_j - x_i\|_2, \quad \text{resp.}$$

In general, the (global) RBF interpolation and approximation errors depend on the fill distance (the smaller the better) as well as the separation distance (the larger the better for stability reasons) of the node set [126, 113, 121], hence these two aspects should be kept in mind for its construction.

We will use the following four prototypes for node distributions in our numerical tests in Section 2.4: regular Cartesian node distributions, a recently introduced node placing algorithm based on Poisson disk sampling which rejects nodes that are too close to each other (according to a user-defined minimal spacing) abbreviated as PNP [113] (created by [115]), Halton node distributions [55, 37] (obtained by the quasi-random numbers of the Halton sequence, see Definition 2.10, created by [18]) and irregular random node distributions (created by [95]).

Definition 2.10. *Let $n = \sum_{i=0}^{k-1} d_i(n) b^i \in \mathbb{N}_0$ be some number in base $b \in \mathbb{N} \setminus \{1\}$ representation with $0 \leq d_i(n) < b$ for all $i \in \{0, \ldots, k-1\}$ for some $k \in \mathbb{N}$.*

1. *The van der Corput sequence $(g_b(n))_{n\in\mathbb{N}}$ in base b is defined by $g_b(n) := \sum_{i=0}^{k-1} d_i(n) b^{-i-1} \in [0, 1)$ [123, 37].*
2. *Let $p(j)$ denote the j'th prime number. The j'th component of the i'th element of the d-dimensional Halton sequence is given by $g_{p(j)}(i)$ for all $i \in \mathbb{N}_0$, $j \in \{1, \ldots, d\}$ [55].*

The construction of these four types of node sets in the interior of our model domain $(0, 1)^d$ is straightforward. The extension of the Cartesian node set to the boundary $\partial\Omega$ is also straighforward whereas different variants exist in the case of PNP, Halton or random nodes, both w. r. t. the number and location of boundary nodes. One could, e. g., use PNP/Halton/random nodes in a lower spatial dimension. We will use the same number of boundary nodes (as determined by the Cartesian grid) for all four types of node sets. The PNP and Halton interior node sets are complemented with boundary nodes of the Cartesian grid whereas the random interior node set is complemented with $d - 1$ dimensional random boundary nodes.

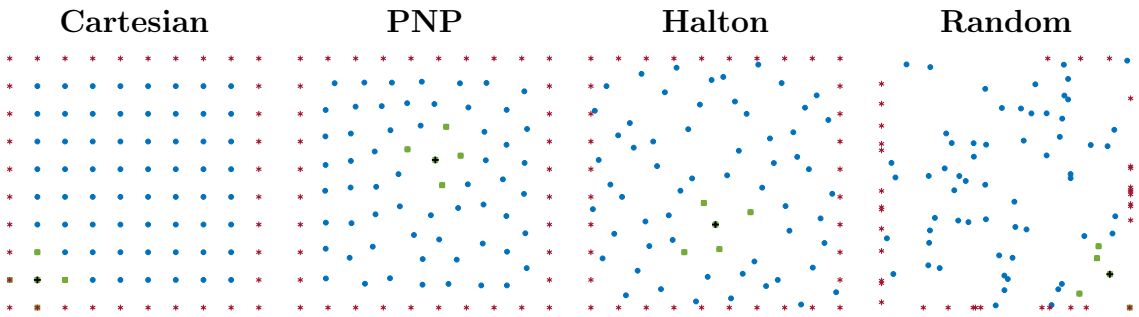

Figure 2.1: Example node distributions and stencils for $\Omega = (0,1)^2$, $N = 100$ (total number of nodes), $N_I = 64$ (number of interior nodes) and $n = 5$ (stencil size).

Figure 2.1 illustrates these four types of node distributions for $d = 2$ spatial dimensions (even though all our numerical experiments will be performed for $d = 3$ spatial dimensions).

2.3.2 Generation of stencils

There are several aspects to take into consideration when selecting the number and location of nodes in a stencil. If a partial differential equation is to be solved, then the stencil will be used to approximate partial/directional derivatives or more general differential operators. Since these describe local characteristics of a function, it is natural to include nodes that are close to the stencil center, e. g. the n nearest neighbors or all neighbors within a certain distance to the center.

Finite difference stencils often have a symmetric structure, i. e., node x_i belongs to stencil X_j if and only if node x_j belongs to stencil X_i (possibly even with identical weights $w_i^j = w_j^i$, e. g. for self-adjoint differential operators). This symmetry property typically no longer holds if upwinding (see Remark 2.11) is used for convection-type operators [110] or if a node lies near the boundary. In this case, one may use an extrapolation approach, introducing new nodes outside the closure $\overline{\Omega}$, or asymmetric stencils [124]. For scattered node sets, we cannot expect any symmetry for RBF-FD stencils.

Remark 2.11. *Upwinding is a strategy (for example for FD and RBF-FD discretizations) to deal with convection-dominated convection-diffusion problems (which are studied in Subsection 2.4.6). The typical idea of upwinding is to suppress non-physical pertubations (i. e., to avoid spurious oscillations) in the numerical solution by taking factors such as the flow direction into consideration during the discretization. This is often done by using different discretizations (e. g., different stencils) for the convection and diffusion terms [19]. [63] proposes to use upwind skewed RBFs, i. e., to multiply the RBFs by an upwind factor which is based on several factors such as the flow direction, the intensity of the convection, the stencil size and the node distribution. Further approaches to use upwinding in RBF-FD are presented in [110, 103, 77].*

In the finite difference method, the stencil size n is typically fixed for all nodes. In the RBF-FD method, it is easily possible (and sometimes advantageous) to utilize different stencil sizes n_j for different stencils X_j, e. g., by selecting a fixed radius and choosing stencils that consist of all nodes within this radius w. r. t. the stencil center [85]. Further propositions and algorithms to form stencils can be found in [23, 91, 24, 105, 92] but are not pursued in more detail here.

Figure 2.1 illustrates nearest neighbor stencils (green squares) for stencil centers (marked with black plus symbols) for the four types of node distributions. The boundary nodes are represented by red asterisk symbols, whereas interior nodes are represented by blue circles if they do not belong to the stencil.

If stencil nodes are very close to each other, the small separation distance often results in high condition numbers of the weight matrices. The criteria for good node distributions for (global) RBF interpolation and approximation as mentioned in Subsection 2.3.1 carry over to the local RBF-FD approach, i. e., a good stencil X_j should consist of a small fill distance (w. r. t. a suitable domain Ω_j containing x_j) and a large separation distance (see Definition 2.9) but does not necessarily have to be structured.

In the case of a structured grid, the weights and resulting truncation errors of RBF-FD may be computed analytically [8, 11]. Stencils may be optimized w. r. t. these truncation errors, e. g. for the discretization of the Laplace operator on Cartesian (tensor product) nodes, we observed that so-called sparse stencils along the coordinate axes can be advantageous compared to nearest neighbor stencils. However, these advantages typically disappear (or are hard to exploit) on unstructured node sets. Stencils on structured grids more likely run into problems with unisolvency (Definition 2.3) than stencils on unstructured grids. A typical example of a node set that is not Π_2-unisolvent in $\mathbb{R}^2$ consists of any three nodes that are collinear, i. e., lying on a line. In [127, Theorem 2.7], a criterion is provided that guarantees for certain (subsets of Cartesian) node sets in $\mathbb{R}^2$ to be Π_ℓ-unisolvent, a generalization to higher spatial dimensions along with additional information on polynomial interpolation in several variables is given in [47, 79]. Here we will pursue only the nearest neighbor approach (with a fixed stencil size for all nodes) and optionally shift (and scale) the stencil afterwards (see Remarks 2.7 and 2.8). For more advanced stencil selection algorithms (which may lead to different conclusions for the choice of RBF-FD parameters) and a machine learning approach to classify the quality of stencils we refer to [22, 23, 25, 91, 99, 92].

If not only the nodes but an underlying, possibly unstructured mesh is available, then a nearest neighbor search could be performed on the graph with the Euclidean distance between nodes being replaced by the shortest path in the graph.

In the case of an unstructured node distribution without a mesh, nearest neighbor stencils may be determined efficiently using a kd-tree [131]. However, especially in the case of randomly distributed nodes, problems may occur if, as illustrated in Figure 2.1 (right), the nearest neighbors are located in a somewhat imbalanced way around the stencil center, possibly with relatively large fill distance and small separation distance. Lop-sided stencils may also occur on structured grids for stencil centers close to the boundary

$\partial\Omega$.

2.3.3 Choice of RBF

Table 2.1 and Figure 2.2 show some of the most commonly used generating functions which can be roughly classified as

- leading to infinitely smooth RBFs and depending on a shape parameter (e. g., GA, GMQ, MQ, IMQ, IQ),
- leading to not infinitely smooth RBFs and (possibly) independent of a shape parameter (e. g., PHS(k)),
- and a hybridization of both (e. g., HYB(γ) = GA + γ PHS(2)) [87, 85].

Additional generating functions can be found in [42, 17], including compactly supported generating functions which lead to a sparse matrix A_j (2.11). Furthermore, there are several modified RBFs such as shifted (see Remark 2.7) and upwind skewed RBFs (see Remark 2.11).

name	abbrev.	$\phi_\varepsilon(r)$
Gaussian (HYB(0))	GA	$e^{-(\varepsilon r)^2}$
Generalized multiquadric	GMQ(ν)	$(1+(\varepsilon r)^2)^{\nu/2},\ \nu \in \mathbb{Z} \setminus 2\mathbb{N}_0$
Multiquadric (GMQ(1))	MQ	$(1+(\varepsilon r)^2)^{1/2}$
Inverse multiquadric (GMQ(-1))	IMQ	$(1+(\varepsilon r)^2)^{-1/2}$
Inverse quadratic (GMQ(-2))	IQ	$(1+(\varepsilon r)^2)^{-1}$
Polyharmonic spline	PHS(k)	r^{2k-1} (or $r^{2k}\log(r)$), $k \in \mathbb{N}$
Hybrid (GA + γ PHS(2))	HYB(γ)	$e^{-(\varepsilon r)^2} + \gamma r^3,\ \gamma \geq 0$ (or $\alpha e^{-(\varepsilon r)^2} + \beta r^3,\ \alpha, \beta\ \in [0,1]$)

Table 2.1: Commonly used generating functions.

For the first class of generating functions, a major challenge in RBF interpolation and approximation (and with that in RBF-FD approximation) is the selection of a suitable shape parameter ε [1]. Choosing it too large results in poor interpolation and approximation accuracies since the RBFs either form peaks (GA, GMQ(ν) with $\nu < 0$, e. g. IMQ, IQ) or approximate piecewise polynomials (GMQ(ν) with $\nu > 0$, e. g. MQ) [29]. On the other side, as the shape parameter ε goes to zero, the RBFs converge to a constant function and the matrix A_j in (2.11) becomes increasingly ill-conditioned, rendering a solution in finite floating point arithmetic numerically infeasible. However, the underlying interpolant may still exist in the limit and can be determined analytically in which case it is related to a multivariate polynomial [45, 104] (which also explains why polynomial augmentation is not needed for small shape parameters [68]). E. g., for $d = 1$, the Lagrange minimal-degree interpolating polynomial can be obtained for the underlying interpolant in the limit $\varepsilon \to 0$. Hence, the FD method can be seen as the limit $\varepsilon \to 0$ in the RBF-FD method [29, 8, 11]. On the other hand, this explains the occurrence of

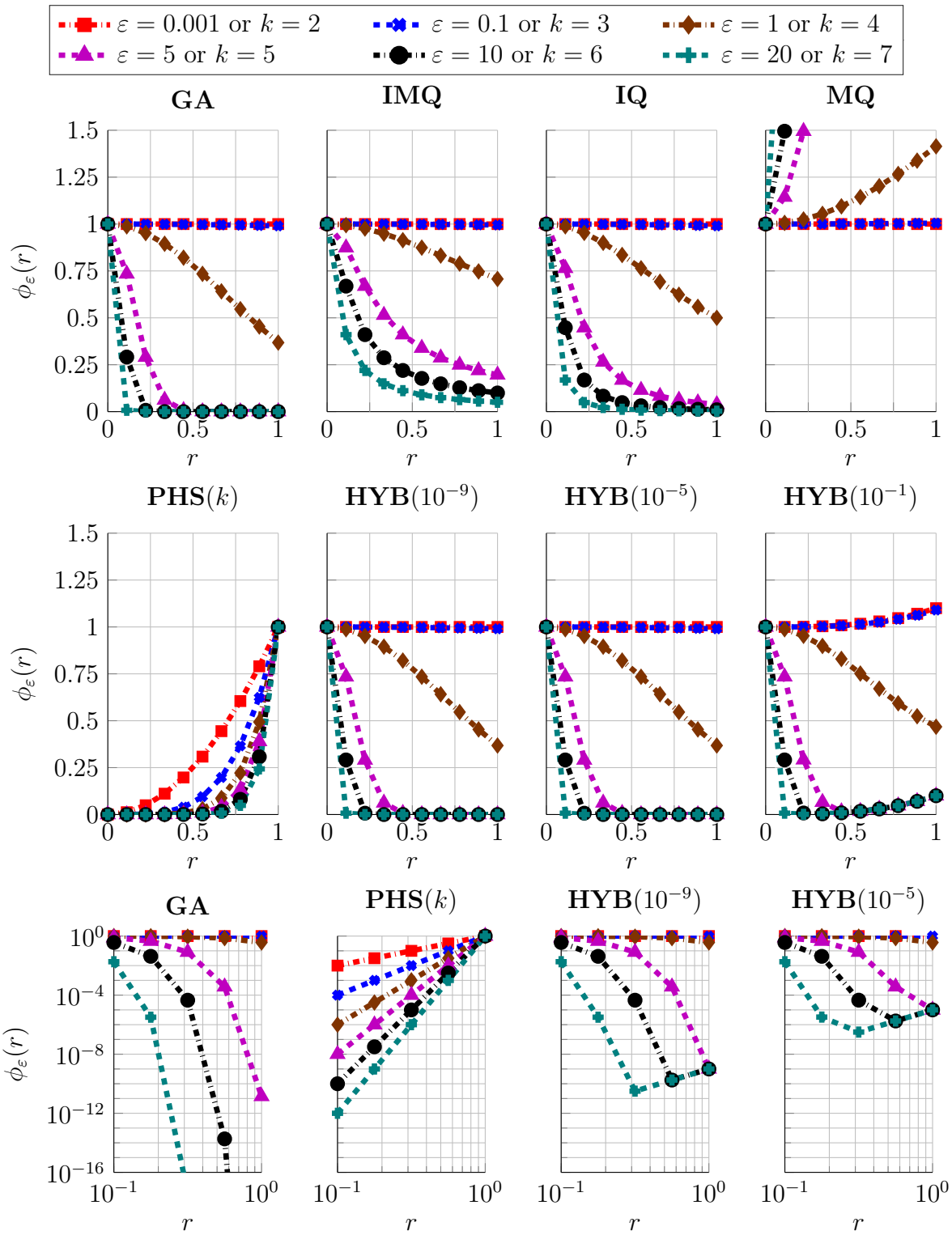

Figure 2.2: Example generating functions for different shape parameters ε or different k, if PHS(k) is shown; last four plots are log-log plots with changed axes.

the Runge phenomenon (i. e., large boundary oscillations in interpolations with higher degree polynomials) in (global) RBF interpolation with small shape parameters. In the context of RBFs, the Runge phenomenon can also be observed if the separation distance becomes very small [46].

For each node set X and stencil X_j, there typically exists a shape parameter $\varepsilon_j > 0$ that is optimal for the numerical computation of the corresponding stencil weights (2.11). Such an ε_j depends on several factors such as the function to be interpolated or approximated, the generating function, the stencil nodes and even the machine precision [67]. Hence, using spatially varying (i. e., stencil dependent) shape parameters ε_j can be (theoretically) advantageous [42, 46, 16], especially for functions with gradient discontinuities [98], but it is not pursued any further here since it adds even more parameters/choices into the setting of RBF-FD. On Cartesian grids, optimal shape parameters can be derived analytically [8, 11].

Another way to address the problem of ill-conditioning is using the hybrid kernel HYB(γ) [87, 85]. The idea is to achieve numerical stability (i. e., lower condition numbers of the weight and stiffness matrices as well as no large spurious eigenvalues of the stiffness matrix for smaller shape parameters) in a direct approach, i. e., with lower computational cost than a stable RBF algorithm. HYB(γ) typically leads to smaller errors and larger condition numbers by decreasing γ and vice versa (i. e., a too small as well as a too large γ value can be problematic in terms of stability or accuracy, resp.). Hence, often

$$\gamma \in [10^{-6}, 10^{-3}] \tag{2.18}$$

is utilized [85, 93].

Yet another (computationally more expensive) possibility to address the ill-conditioning is to switch to stable versions of RBFs, e. g., RBF-GA [44, 76, 71], RBF-RA [48, 129] or RBF-QR [68]. These stable algorithms typically increase the computational cost by at least a factor of 10 [106]. Using (sufficiently high) extended precision is another approach, but in general not advisable for large scale problems since it is computationally even more expensive [41, 68].

A feature of the PHS generating function is illustrated in Remarks 2.12 and 2.14, namely the scale-invariance of PHS (in addition to the shift-invariance of RBFs, see Remark 2.7).

Remark 2.12. *In [57], the following results are shown for the PHS generating function.*

1. *The Lagrange representation (with Lagrange basis functions λ_i, $i \in \{1, \dots, n\}$) of $s \in \mathcal{R}_j$ (2.7) to solve (2.8) (w. r. t. the stencil X_j) is given by*

$$s(x) = \sum_{i=1}^{n} \lambda_i(x) f(x_{s_i^j}). \tag{2.19}$$

2. *The Lagrange basis functions λ_i, $i \in \{1, \dots, n\}$ are shift and scale invariant. Hence, the Lagrange representation of $s \in \mathcal{R}_j$ to solve (2.8) w. r. t. the shifted*

and scaled stencil $X_j^{h_j}$ is identical to (2.19) *except that the evalution points of the function f are given by the stencil $X_j^{h_j}$ (instead of the stencil X_j).*

Definition 2.13. *We call (according to [26])*

1. *the domain Ω scalable if the origin is an interior point and $h\Omega \subseteq \Omega$ holds for all $h \in [0,1]$;*
2. *the function space $\mathcal{R}_j$* (2.7) *scalable if $s(h\cdot) \in \mathcal{R}_j$ holds for all $h \in (0,1]$, $s \in \mathcal{R}_j$.*
3. *The differential operator $\mathcal{L}$ applied to a function $s \in \mathcal{R}_j$ of a scalable function space $\mathcal{R}_j$ has scaling order a if $\mathcal{L}s(h\cdot) = h^a \mathcal{L}s$ holds for all functions $s \in \mathcal{R}_j$.*
4. *The Approximation* (2.3) *is called scalable of order a if the domain Ω is scalable, the function space $\mathcal{R}_j$ is scalable and the differential operator $\mathcal{L}$ applied to a function $s \in \mathcal{R}_j$ has scaling order a.*

The combination of Remarks 2.7 and 2.12 leads to Remark 2.14.

Remark 2.14. *It holds [26]:*

1. *If the stencil X_j is replaced by the shifted and scaled stencil $X_j^{h_j}$ (and the shifted evaluation point is used, i. e., the origin instead of x_j), the PHS generating function is utilized and the Approximation* (2.3) *is scalable of order a, then the weights w^j* (2.11) *and $w^j_{h_j}$ corresponding to the stencils X_j and $X_j^{h_j}$, resp., fulfill*

$$w^j = h_j^{-a} w^j_{h_j}.$$

2. *The scaling order a is for linear homogeneous differential operators $\mathcal{L}$ (such as the Laplace operator) given by the order D of the differential operator $\mathcal{L}$ (i. e., $a = D = 2$ for the Laplace operator).*

2.3.4 Polynomial augmentation in RBF-FD

Polynomial interpolation can be viewed as a special case of RBF-FD with polynomial augmentation in which the dimension of the polynomial space M equals the stencil size n, but no radial basis functions are used, i. e., the stencil weights are computed as the solution of $P_j^T w_j = \mathcal{L}p_j$ (2.11). As mentioned in Subsection 2.2.1, polynomial interpolation can lead to a singular P_j if $\min\{d, n\} > 1$. If polynomial augmentation is used in RBF-FD, necessary (but not sufficient) criteria for the weight matrix $\widetilde{A}_j$ in (2.11) to be regular are given by $n \geq M$ and $\operatorname{rank}(P_j) = M$ (full column rank).

Often, for a numerically stable computation, a lower bound $n_{\min} = n_{\min}(\ell)$ is suggested for the stencil size n that depends on the degree ℓ of polynomial augmentation (or, equivalently, the dimension M of the respective polynomial space (2.5)). In [41], the lower bound $n_{\min} = M$ is suggested for RBF-FD approximation. For the solution of

elliptic and hyperbolic PDEs, $n_{\min} \approx 2M$ has been proposed [10, 38, 61]. A slightly different bound, $n_{\min} \approx 2M + \lfloor \ln(2M) \rfloor$ is proposed in [107] for the advection-diffusion equation. In [38, 102], it is observed that unstructured node distributions allow for smaller $n_{\min}$ than Cartesian node distributions with an increasing difference as ℓ increases (e. g., $n_{\min} \approx 2M$ is too small for Cartesian nodes with $\ell > 4$ whereas unstructured nodes can achieve lowest errors for $n \approx M + \lfloor \ln(2M) \rfloor$ [102]). In general, the optimal relation between polynomial degree ℓ and stencil size n depends on the underlying problem [107].

There are different effects of polynomial augmentation in the RBF-FD method which depend on the type of generating function that they complement. Polynomial augmentation (with close too maximal permissible degree ℓ) is studied in [41, 42] for RBF-FD approximation for the GA generating function (with a shape parameter ε, leading to infinitely smooth RBFs) and the PHS(k) generating function (without a shape parameter, leading to not infinitely smooth RBFs). In both cases, stagnation errors disappear and higher accuracies (for the same convergence order) compared to polynomial least squares approximation with the same ℓ are observed (for sufficiently, but not too large ℓ [121]). The convergence order for $\ell \geq D$ (2.20) is then

$$\mathcal{O}(N^{(D-\ell-1)/d}), \quad D = \text{ order of the linear differential operator } \mathcal{L} \tag{2.20}$$

if the approximated function is sufficiently smooth [108, 121] (i. e., $D = 0$ for function approximation and $D = 2$ for the Laplace operator $\mathcal{L} = \Delta$ or the convection-diffusion operator $\mathcal{L} = \Delta + b^T \nabla$) and (approximate) internodal spacing $h \approx N^{-1/d}$ [41, 68, 6] (see (2.17) for the smallest achievable error w. r. t. floating point arithmetic). Hence, the convergence order depends on ℓ (which is connected to n as a larger ℓ requires a larger n). While n, ε and k do not enter into the above order of convergence, varying these parameters can still lead to significant changes in the accuracy [41], i. e., affect the constants hidden in the $\mathcal{O}$ notation.

For GA, the accuracy can be significantly increased by polynomial augmentation for shape parameters larger than the optimal one [44] (while possibly increasing the condition number of $\widetilde{A}_j$ in (2.11)) and the optimal shape parameter is (similar to the case without polynomial augmentation) mostly invariant w. r. t. refinement (i. e., the number of nodes N).

For PHS (or other only conditionally positive definite RBFs), polynomial augmentation is typically needed for convergence [85, 61]. The condition number of the matrix $\widetilde{A}_j$ in (2.11) may increase (also for PHS) when increasing the polynomial degree ℓ. Furthermore, when increasing the stencil size n, the stencil weights become larger (by absolute value) near the stencil center and smaller elsewhere, and if the stencil size n is large enough, then there is no Runge phenomenon [7, 6].

Infinitely smooth RBFs have the potential for spectral accuracy in global RBF approaches [46], but this benefit is typically sacrificed in the local RBF-FD approach [11] since the stencil size n is significantly smaller than the number of nodes N. Additionally, GA and PHS with polynomial augmentation (and n only slightly larger than M) can achieve (especially for unstructured nodes since they allow for smaller $n_{\min}$ than Carte-

sian nodes [102]) comparable accuracies with lower computational cost than stable RBF variants. Using the GA generating function (with a stable algorithm or close to maximal permissible degree of polynomial augmentation) leads to the challenge of choosing an appropriate shape parameter, and using extended precision is computationally too expensive for RBF-FD approximation. Hence, the PHS generating function with close to maximal permissible degree of polynomial augmentation is recommended (taking into consideration the approximation accuracy and computational cost) [41, 42].

Many results for RBF-FD approximation using PHS with polynomial augmentation carry over to the solution of elliptic and hyperbolic PDEs, i. e., stagnation errors disappear, the convergence order is again given by (2.20), hence depends on the polynomial degree ℓ (which requires a minimal stencil size $n_{\min}$). Increasing the PHS degree determined by k and stencil size n may lead to slight improvements of accuracy. The stencil weights become larger (by absolute value) near the stencil center and smaller elsewhere by increasing the stencil size n. If the stencil size n is large enough, then there is no Runge phenomenon [10, 38].

For the Poisson problem (involving a differential operator $\mathcal{L}$ of order $D = 2$ (2.20)), a convergence order of $\mathcal{O}(N^{-\ell/d})$ is numerically observed for $\ell \in \{2, 4, 6, 8\}$ (and no convergence for $\ell = 0$) [61], i. e., slightly higher than the (theoretical) convergence order for RBF-FD Approximation (2.20). Additionally, it is observed in [10, 38] that using PHS in the form r^{2k-1} or $r^{2k}\log(r)$ leads to similar results (thus the form r^{2k-1} is preferred), quasi-uniform node distributions lead to similar (often even higher) accuracies compared to Cartesian node distributions, and if the stencil size n is sufficiently large, then no ghost nodes (i. e., additional nodes outside the domain Ω to increase stability and accuracy by avoiding lop-sided stencils near the boundary $\partial\Omega$) are needed. It has been observed in [121] that ghost nodes can increase stability and accuracy especially in the presence of Neumann boundary conditions. Further stabilization techniques (which are based on modifications of the boundary nodes) for problems with Neumann boundary conditions are introduced in [132]. Moreover, whereas a larger PHS degree can lead to a higher accuracy (accompanied by the potential for more numerical instabilities such as larger spurious eigenvalues of the stiffness matrix and larger absolute values of the stencil weights), a PHS degree that is chosen too small can also be problematic. Hence, given the degree of augmentation $\ell \in \mathbb{N}$, in [107] it is suggested to select k in the PHS r^{2k-1} according to

$$k = \begin{cases} \frac{\ell+1}{2}, & \text{if } \ell \text{ is odd,} \\ \frac{\ell}{2}, & \text{if } \ell \text{ is even.} \end{cases} \tag{2.21}$$

While the optimal choice for the relation between PHS exponent k and polynomial degree ℓ could depend on the type of the node distribution, the relation

$$k \leq k_{\max} := \ell + 1 \tag{2.22}$$

has to be satisfied to prove unisolvency in this setting [107]. Furthermore, there has to

hold

$$D \leq k \tag{2.23}$$

for the order D of the differential operator (2.20) to ensure that the right hand side $\mathcal{L}\Phi_{\varepsilon,x_{s_i^j}}(x_j)$ in (2.11) is defined for all stencil nodes $x_{s_i^j}$, in particular for the center $s_i^j = j$. Since (2.21) contradicts (2.23) for the case $\ell = 2$, we modify (2.21) and define (with D as in (2.20))

$$k_{\min} := \begin{cases} \max\{D, \frac{\ell+1}{2}\}, & \text{if } \ell \text{ is odd,} \\ \max\{D, \frac{\ell}{2}\}, & \text{if } \ell \text{ is even} \end{cases} \tag{2.24}$$

and propose to select $k \in [k_{\min}, k_{\max}]$ depending on the model problem and number of points N (see also (2.26) for our choice in the numerical experiments).

A different idea lies in the use of variable degrees of polynomial augmentation ℓ_j and variable sizes n_j for each stencil X_j [86, 111, 92, 62, 59, 23, 91]. While it can reduce the computational cost and increase the accuracy, it adds even more parameters/choices into the setup of RBF-FD, hence this approach is not pursued any further here.

2.4 Numerical results

This section contains numerical tests first for the RBF-FD discretization of Poisson's equation in $d = 3$ spatial dimensions on a unit cube, i. e., in (2.12a), (2.12b), we set $\Omega = (0,1)^3$ and $\mathcal{L} = -\Delta$. More numerical tests for further model problems are afterwards presented in Subsections 2.4.5 and 2.4.6. We choose the right hand side functions f, g such that the analytical solution u is given by the three-dimensional version of Franke's function $F\colon \mathbb{R}^3 \to \mathbb{R}$ [96, 70] with

$$\begin{aligned} F(x) = 0.75\exp[-\left((9x_1-2)^2 + (9x_2-2)^2 + (9x_3-2)^2\right)/4] \\ +0.75\exp[-(9x_1+1)^2/49 - (9x_2+1)/10 - (9x_3+1)/10] \\ +0.5\exp[-\left((9x_1-7)^2 + (9x_2-3)^2 + (9x_3-5)^2\right)/4] \\ -0.2\exp[-(9x_1-4)^2 - (9x_2-7)^2 - (9x_3-5)^2]. \end{aligned} \tag{2.25}$$

We will study the influence of the following parameters on the performance of the RBF-FD method to illustrate causes for potential pitfalls and hopefully steer away from them:

- the type of the node distribution (i. e., Cartesian, PNP, Halton or random);
- the stencil size n;
- the total number of nodes N;
- the generating function ϕ_ε and its intrinsic parameters (shape parameter ε, the parameter γ in HYB(γ), the PHS degree specified by k);

- the degree of polynomial augmentation ℓ.

We will illustrate the resulting approximation errors of the discrete approximate solution side-by-side with the condition numbers of the involved weight and stiffness matrices. To this end, all of the following Figures 2.4 - 2.19 consist of three columns of plots, showing

- the errors between the exact and computed solution of the PDE (left column),
- the (maximum of the) condition numbers of the weight matrices (2.11) (middle column),
- the condition numbers of the stiffness matrices (2.14), (2.15) (right column).

Figures 2.20 and 2.23 - 2.25 show solely the errors, whereas Figure 2.21 concentrates on the condition numbers of the stiffness matrices. The y-axes in the figures in this chapter as well as in the next chapter are usually chosen such that

- comparable plots in a figure use identical y-axes to simplify the comparison of different setups;
- the amount of unnecessary whitespace is minimized;
- no relative errors greater than one are shown (since the zero vector already leads to a relative error of one).

However, outliers in a plot are sometimes cut off to increase the distinguishability and the visibility of the other results in this plot.

In Subsection 2.4.1, we present general information about the setting of the numerical tests such as node and stencil generation, solution of the arising linear systems of equations and the computation of the approximation error. In Subsection 2.4.2, we focus on the pitfalls for infinitely smooth (GA) RBFs related to the choice of the shape parameter. We concentrate in Subsection 2.4.3 on the (shape parameter independent) PHS generating function. In Subsection 2.4.4, we utilize the HYB generating function in the same setups as in Subsection 2.4.2 and compare HYB to infinitely smooth RBFs as well as to PHS. Subsections 2.4.5 and 2.4.6 show more numerical tests for other problems, namely the sphere instead of the cube as the underlying domain and the convection-diffusion equation instead of Poisson's equation, resp. The conclusions are summarized in Subsection 2.4.7, along with recommendations for the various components of RBF-FD discretization.

2.4.1 General information

In order to compare results for the different types of node distributions (see Subsection 2.3.1), we generate PNP, Halton and random node distributions with the same number of interior and boundary nodes as in a typical Cartesian grid, i. e., $N_I = \nu^d$ interior nodes and $N = (\nu + 2)^d$ total nodes for some $\nu \in \mathbb{N}$. While the construction of Halton or random nodes with a prescribed number N_I of interior nodes is straighforward, it

is slightly more complicated for PNP nodes since their construction is based on a local distance function $h\colon \Omega \to (0,\infty)$ that specifies the minimal nodal spacing at point $x \in \Omega$ via $h(x)$. A constant nodal spacing function $h \equiv (\nu+1)^{-1}$ (i. e., the same spacing as for the Cartesian nodes) leads to fewer nodes than N_I. Hence, we use a scaled version $h \equiv \alpha(\nu+1)^{-1}$ with $\alpha \in [0.9256, 0.9286]$ experimentally determined such that exactly N_I nodes are generated. (The tests with $h \equiv (\nu+1)^{-1}$ lead to qualitatively comparable observations as $h \equiv \alpha(\nu+1)^{-1}$ with slightly increased errors and decreased condition numbers due to the typically larger fill and separation distances).

When showing results on Cartesian grids in Subsections 2.4.2 - 2.4.4, for comparison we also include test results (for the errors (2.27) and the condition numbers of the stiffness matrices) obtained for finite difference discretizations with a standard central 7-point stencil (i. e., second-order approximation) as well as for a compact/Hermite 19-point stencil (i. e., the 19 stencil nodes consist of the $3 \times 3 \times 3$ nodes around the center without the 8 corner nodes, and not only function values but also derivatives are included in the approximation in (2.3), leading to a fourth-order approximation, see [128], [116]). The results on Cartesian grids in Subsection 2.4.6 show, for comparison, solely results obtained for finite difference discretizations with a standard central 7-point stencil without upwinding (i. e., no results for compact/Hermite 19-point stencil are included).

For the computation of stencils (see Subsection 2.3.2), we use a library [89] that utilizes a kd-tree for the nearest neighbor search w. r. t. the norm $\|\cdot\|_\mu$ for some $\mu \in \mathbb{N}$. In our experiments, we use the Euclidean norm by setting $\mu = 2$. For the computation of the stencil weights (2.11), we use the monomial basis $\{p_1, \dots, p_M\} = \{p_1^m, \dots, p_M^m\}$ of Π_ℓ (2.5) (if not stated otherwise).

We focus on the generating functions in Table 2.1, i. e., we consider the generating functions GA, MQ, IMQ, IQ, HYB(γ) and PHS(k) (which requires polynomial augmentation of degree $\ell \geq 2$ for convergence, see (2.20)) of the form r^{2k-1} with $k \geq 2$ usually determined by

$$k = \begin{cases} \ell+1, & \text{if } \ell \leq 3, \\ \ell, & \text{if } \ell \geq 4 \end{cases} \tag{2.26}$$

or by (2.24).

Even though some generating functions (e. g., GA, IMQ, IQ [42]) guarantee the resulting matrices A_j in (2.11) to be symmetric positive definite, they might be highly ill-conditioned so that a direct solver based on a Cholesky factorization may break down in practice. Hence, for solving linear systems, we will always use direct solvers based on LU factorizations with pivoting. For stencils that are not unisolvent, the matrix block P_j in (2.11) may no longer have full column rank, hence $\widetilde{A}_j$ is no longer regular but (2.11) may still have (no longer unique) solutions. These can be computed by a nullspace approach [21, 24] which may also be beneficial in the unisolvent case. For the computation of (lower bounds of) condition numbers κ_1 of the arising matrices, we use the LAPACK [66] function dgecon with the 1-norm (we also computed κ_∞ which led to

qualitatively similar observations for the stiffness matrix and made no difference for the symmetric weight matrices, hence these results are not reported here).

In order to measure the approximation accuracy of the RBF-FD computed discrete solution, we determine the relative 2-norm error between the exact solution of the PDE (2.12a), (2.12b) evaluated at the nodes, $u_s := (u(x_j))_{j=1}^{N_I}$, and the computed solution u obtained from solving the system in (2.15),

$$e_2 := \frac{\|u_s - u\|_2}{\|u_s\|_2}. \tag{2.27}$$

We have also performed numerical experiments and measured the errors in the (relative) infinity as well as the (relative) 1-norm but do not report these results here since they are qualitatively very similar (see also [61, 30] for results in different error norms). All computations were performed in double precision.

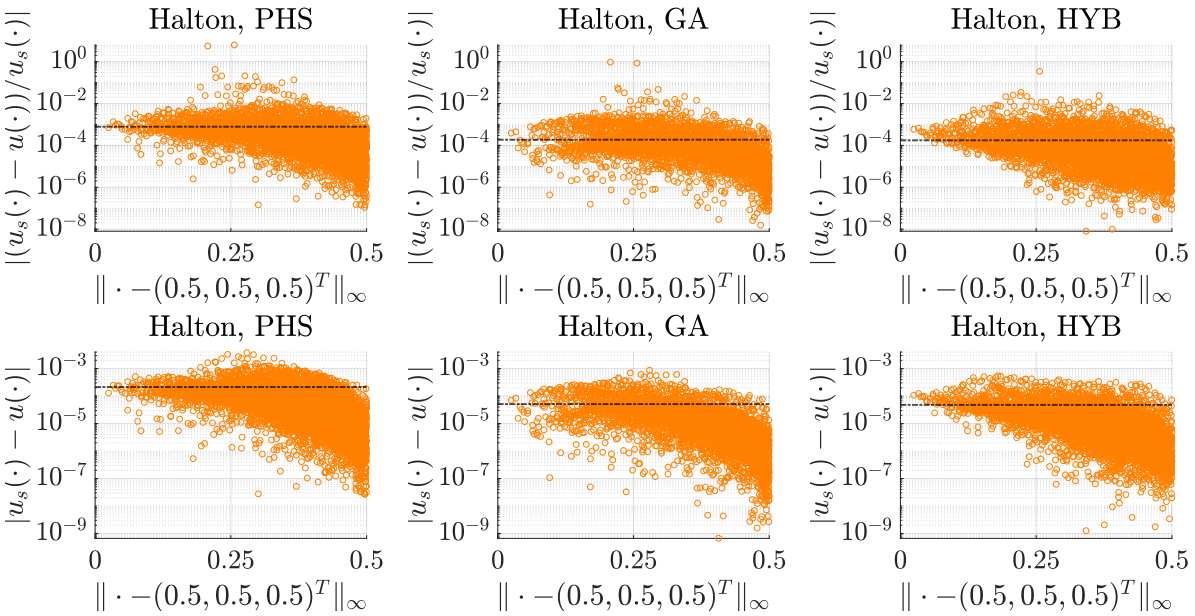

Figure 2.3: Pointwise relative errors $|(u_s(\cdot) - u(\cdot))/u_s(\cdot)|$ (top row) and absolute errors $|u_s(\cdot) - u(\cdot)|$ (bottom row) w. r. t. their $\|\cdot\|_\infty$ norm distance to the center $(0.5, 0.5, 0.5)^T$ of the domain $[0, 1]^3$ for Halton node distributions with $N = 8000$ nodes and $N_I = 5832$ interior nodes. PHS with stencil size $n = 116$, polynomial augmentation of degree $\ell = 5$; GA and HYB(10^{-5}) with $n = 131$, shape parameter $\varepsilon = 3.5$, no polynomial augmentation.

Since the error e_2 in (2.15) does not provide insight into the component-wise absolute or relative errors $|(u_s(\cdot) - u(\cdot))|$ or $|(u_s(\cdot) - u(\cdot))/u_s(\cdot)|$, resp., we include Figure 2.3 which illustrates these pointwise errors w. r. t. their distance to the center $(0.5, 0.5, 0.5)^T$ of the domain $[0, 1]^3$ for three different types of generating functions on a Halton grid. While we observe that the pointwise errors possess a significant deviation from their

mean and tend to become smaller closer to the (Dirichlet) boundary, we argue that the error e_2 (added as a horizontal line to all plots, scaled by $1/\sqrt{N_I}$ for the bottom row) is a reasonable measure to assess the overall RBF-FD discretization error.

2.4.2 (Infinitely smooth) Gaussian (GA) RBFs

In this subsection, we provide numerical tests using the GA generating function which leads to infinitely smooth RBFs. We have also performed the same tests for generalized multiquadric GMQ(ν) for $\nu \in \{1, -1, -2\}$ (i. e., for MQ, IMQ and IQ) generating functions but do not report on these results since they were qualitatively similar (with adjusted shape parameters and constants c in (2.28)).

Similarly, since random nodes lead to qualitatively comparable observations as Halton nodes (with slightly increased errors and condition numbers due to the typically larger fill distance and smaller separation distance), we also do not include our test results for random nodes here.

We will later (in Subsection 2.4.4) provide numerical results comparing several generating functions, including MQ, IMQ and IQ, on all four types of grids (Cartesian, PNP, Halton, random).

We begin with an overview of the numerical tests before a more detailed discussion of the results.

- Figure 2.4: Influence of the shape parameter ε on different grid types (Cartesian, PNP, Halton) for different stencil sizes n, fixed problem size N.
- Figure 2.5: Influence of the stencil size n on different grid types (Cartesian, PNP, Halton) for different shape parameters ε, fixed problem size N, polynomial augmentation of degree $\ell = 0$ (a constant term).
- Figure 2.6: Influence of the problem size N on different grid types (Cartesian, PNP, Halton) for different stencil sizes n, a problem size dependent shape parameter ε (2.28), polynomial augmentation of degree $\ell = 0$.

In Figure 2.4, we show results w. r. t. the shape parameter $\varepsilon \in \{0.2m : m = 1, 2, \ldots, 30\}$ for the GA generating function, stencil sizes $n \in \{7, 35, 50, 90, 131\}$ and node distributions (Cartesian, PNP, Halton) with $N = 20^3 = 8000$ total nodes ($N_I = 18^3 = 5832$ interior and $N - N_I = 2168$ boundary nodes). Each color/marker represents a different stencil size.

In the three left plots, we see that there exists an optimal shape parameter $\varepsilon \in (1, 4)$ that increases with increasing stencil size, especially on the Cartesian and PNP grids. For (much) larger shape parameters, the discretization errors increase since the basis functions turn more and more into spikes so that the spanned function space does not offer accurate approximations. For (much) smaller shape parameters, numerical instabilities occur which become apparent in the middle and right plots showing condition numbers for both the weight and stiffness matrices. Larger stencil sizes incur larger

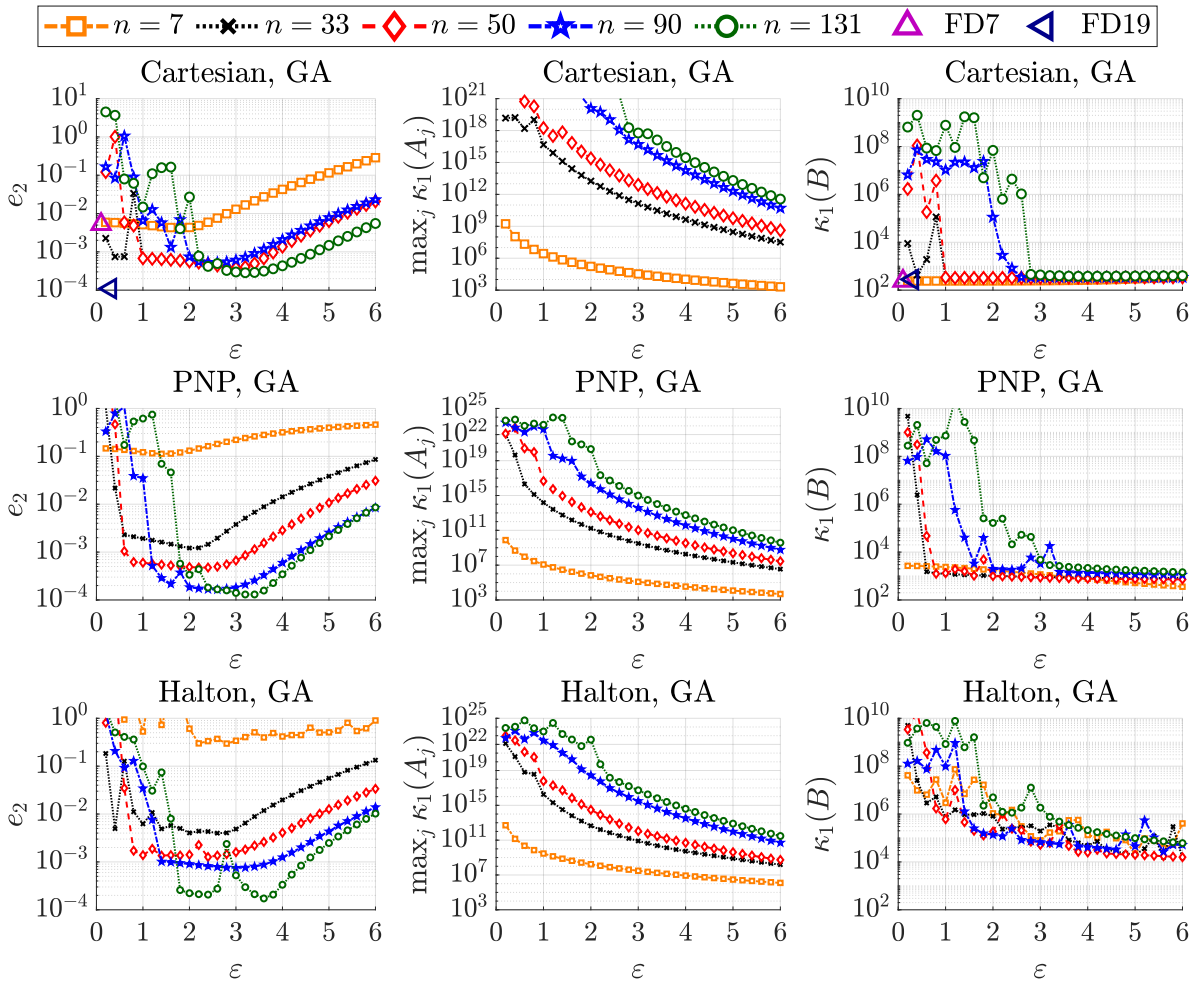

Figure 2.4: Errors (2.27) and condition numbers as a function of the shape parameter ε for the GA generating function, stencil sizes $n \in \{7, 35, 50, 90, 131\}$ and node distributions (Cartesian, PNP, Halton) with $N = 8000$ nodes and $N_I = 5832$ interior nodes.

condition numbers in the weight matrices but also have the potential to reach smaller discretization accuracies for suitably chosen shape parameters. In particular, the (best possible) RBF-FD discretization accuracies obtained on the PNP and Halton grids are comparable to the ones on the Cartesian grid (or even better) as long as the stencil size is large enough.

In Figure 2.5, the setting is similar to the one in Figure 2.4, only that now the stencil size $n \in \{7, \ldots, 200\}$ (whereas only every fifth marker is visible to avoid too cluttered plots) is shown along the x-axis and each color/marker represents a different shape parameter $\varepsilon \in \{1, 2, 3, 4, 5\}$. Furthermore, polynomial augmentation of degree $\ell = 0$ (i. e., a constant term) has been added. This choice typically increases the accuracy slightly with only marginal increase in the computational cost (see Subsection 2.3.4). This figure once more illustrates the numerical instabilities that occur for (too) small

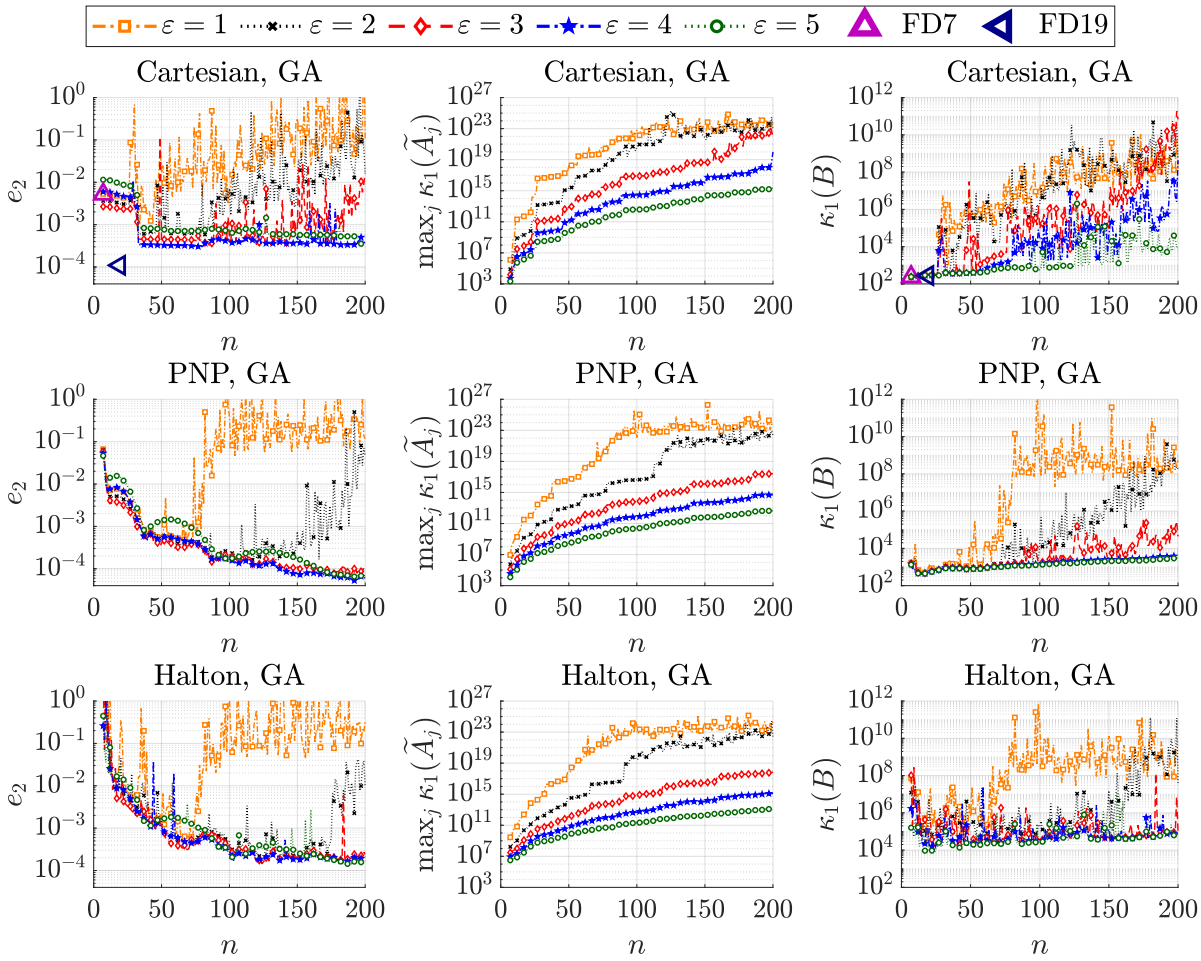

Figure 2.5: Errors (2.27) and condition numbers as a function of the stencil size n for the GA generating function, shape parameters $\varepsilon \in \{1, 2, 3, 4, 5\}$, polynomial augmentation of degree $\ell = 0$ and node distributions (Cartesian, PNP, Halton) with $N = 8000$ nodes and $N_I = 5832$ interior nodes.

shape parameters. On the Cartesian grid, one can observe a noticeable jump that occurs when increasing the stencil size from $n \leq 32$ to $n \geq 33$ which was also observed in [11] and coincides with the following observation: The stencil nodes that are most valuable for the approximation of the Laplace operator on the Cartesian grid appear to be those along the coordinate axes. The first 27 nodes in a nearest neighbor search are those in the $3 \times 3 \times 3$ grid around the stencil center, the next 6 are the extensions to both sides along the respective three coordinate axes. When all 6 are added and a stencil of 33 nodes is formed, one sees the jumping improvement in the approximation accuracy. On the Cartesian grid, a further increase in the stencil size leads to hardly any improvement. On the Halton grid, stagnation sets in for a stencil size around $n = 100$ whereas the errors continue to (slowly) decrease up to a stencil size around $n = 150$.

In Figure 2.6, we show plots w. r. t. $\sqrt[3]{N}$ where $N \in \{(\nu + 2)^3 : \nu \in \mathbb{N}, 6 \leq \nu \leq 25\}$

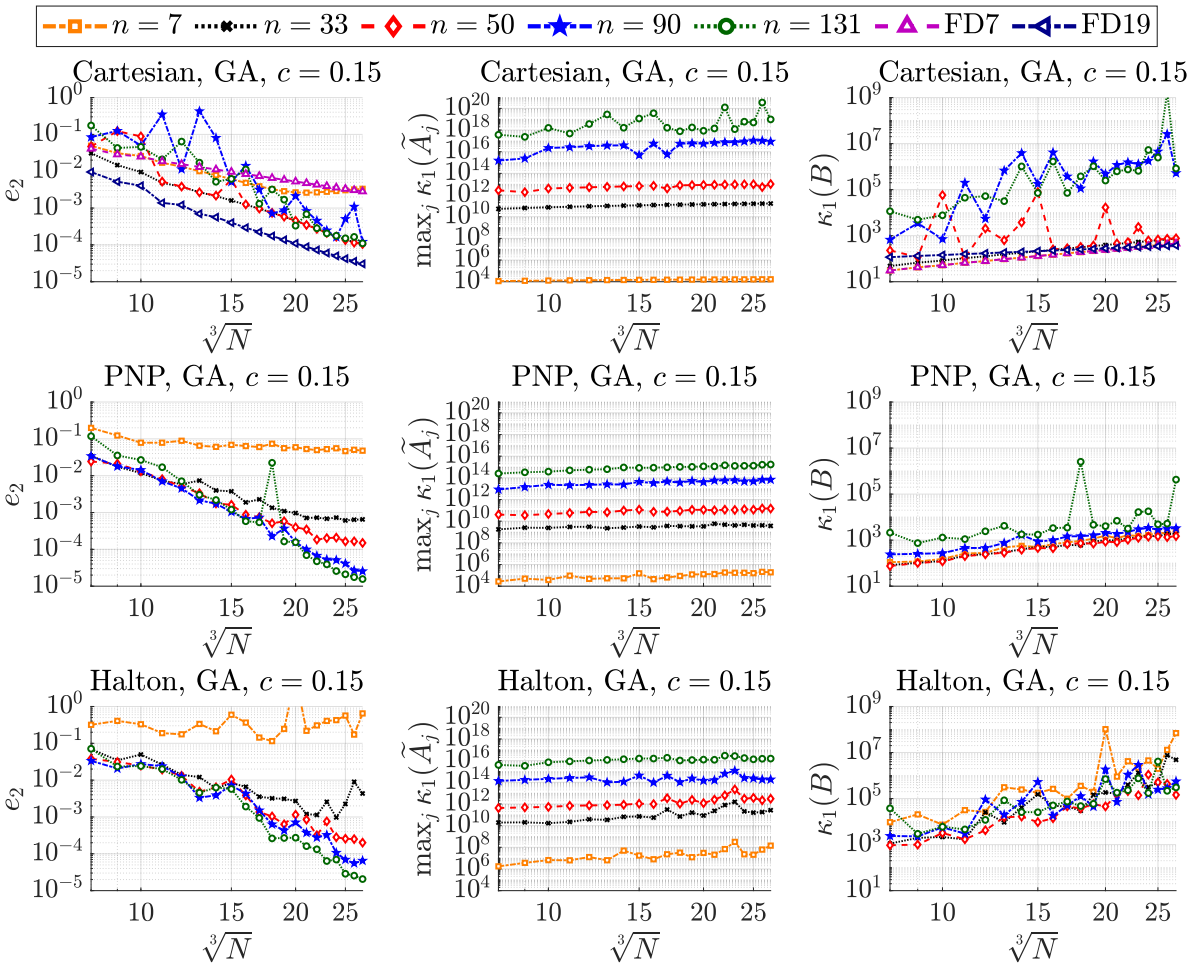

Figure 2.6: Errors (2.27) and condition numbers as a function of the number of nodes for the GA generating function with polynomial augmentation of degree $\ell = 0$, $c = 0.15$ (2.28) and stencil sizes $n \in \{7, 33, 50, 90, 131\}$.

denotes the number of total nodes with $N_I = \nu^3 \in [216, 15625]$ interior nodes. As before, we use the GA generating function with polynomial augmentation of degree $\ell = 0$. For the shape parameter, we follow the strategy to increase it along with the problem size as in

$$\varepsilon := cN^{1/d} \tag{2.28}$$

for a constant c to avoid increasing ill-conditioning in the weight matrices since the consequences of $\varepsilon \to 0$ for fixed N and of $N \to \infty$ for fixed ε are identical [38, 40, 85, 118]. The constant c influences the shape parameter, a smaller c leads to larger condition numbers of the weight (stiffness) matrices, a larger c leads to "spiky" basis functions that no longer provide good approximations on finer grids. A similar approach using two constants c_1, c_2 with $\varepsilon = c_1 N^{1/d} - c_2 > 0$ (and the determination of these constants) has been studied in [40].

We have performed several tests for constants $c \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$ (not reported here) and found $c = 0.15$ to yield the best results for a stencil size $n = 50$ (and good results for a wider range of stencil sizes). In the plots in the middle of Figure 2.6, using (2.28) to determine the shape parameter, it is confirmed that with such a choice of shape parameter, the condition numbers of the weight matrices remain (close to) constant (with different constants for different c) as N increases while the condition numbers of the stiffness matrix still increase with N.

On a Cartesian grid, the RBF-FD errors for a stencil of size $n = 7$ are comparable to the 7-point-FD errors (even better up to $\sqrt[3]{N} \approx 25$ and worse for larger N since $c = 0.15$ is too large for this n), and stencil sizes $n > 33$ appear to lead to no further advantage compared to $n = 33$. On PNP and Halton grids, RBF-FD with stencil size $n = 7$ performs worse than on a Cartesian grid. For larger stencils, the errors decrease both with increasing stencil and problem sizes n, N, reaching accuracies even better than the finite difference 19-point-stencil.

2.4.3 Polyharmonic spline (PHS) RBFs

An attractive feature of the PHS generating function with polynomial augmentation lies in their simplicity due to the lack of a shape parameter ε to be finetuned and their scalability (see Remark 2.14), i. e., the stencil weights w^j (2.11) can be recovered from the solution $w^j_{h_j}$ of a shifted and scaled (and typically much better conditioned) linear system of equations. Since the high condition numbers in (2.11) hence can be reduced by a (row and column) scaling, they are usually harmless in actual computations and have no negative effect on the computed interpolant [41] as long as the degree of polynomial augmentation is not too large [121]. We will illustrate the (typical lack of) difference between the numerical solutions with and without shift-and-scale and with only shift in Figure 2.9 and use Remark 2.14 instead of (2.11) in all other following numerical tests for Poisson's equation involving PHS (if not stated otherwise).

However, even without a shape parameter, there still remain several parameters to choose such as the stencil size n, the exponent in the polyharmonic spline determined by k and the degree ℓ of polynomial augmentation. Similarly as in Subsection 2.4.2, random nodes lead to worse, but qualitatively comparable observations as Halton nodes. Hence, we do not include test results for random nodes (Subsection 2.4.4 will include results for PHS on random nodes in comparison to other RBFs). In this subsection, we provide the following numerical results to illustrate the interplay of these quantities and their resulting discretization errors.

- Figure 2.7: Influence of the stencil size n on different grid types (Cartesian, PNP, Halton) for the PHS(k) generating function and different degrees ℓ of polynomial augmentation where k is determined by ℓ via (2.24) or (2.26).
- Figure 2.8: Influence of the problem size N on different grid types (Cartesian, PNP, Halton) for different k in PHS(k) and degrees ℓ of polynomial augmentation

which in turn determine the stencil size n via (2.29).

- Figure 2.9 (as Figure 2.8, with solely Halton nodes and degree $\ell = 8$ of polynomial augmentation): Influence of shift (Remark 2.7), shift-and-scale (Remark 2.14) and compiler optimization.
- Figure 2.10: Influence of the problem size N on different grid types (Cartesian, PNP, Halton) for different degrees ℓ of polynomial augmentation which in turn determine k in PHS(k) and the stencil size n via (2.26) and (2.29), resp.

In Figure 2.7, each color/marker represents a different degree of polynomial augmentation $\ell \in \{2, 3, 4, 5, 6, 7, 8\}$. For a given ℓ, we use the generating function PHS(k) with k selected via (2.24) (odd rows) and via (2.26) (even rows) and use stencil sizes $n \in \{2M + \lfloor 2M/15 \rfloor i \ : \ i \in \{-4, \dots, 17\}\} \subset [\lfloor 22M/15 \rfloor, \lfloor 64M/15 \rfloor]$ (where M (2.6) denotes the dimension of the space of augmented polynomials, see Table 2.2).

For Cartesian nodes and $\ell \in \{2, 3, 5\}$, there occured breakdowns in the solver (i. e., in the LU factorization of the weight matrices) for some of the smaller stencil sizes. The nullspace approach of [21, 24] could have been used instead of an LU factorization to compute (non-unique) weights but was not pursued here. The second column of Table 2.2 summarizes (lower bounds for) the minimal stencil sizes $n_{\min}$ which were necessary for the Cartesian grid in this setup. Larger degrees $\ell \geq 6$ appear to require even larger stencils than $\lfloor 64M/15 \rfloor$. In the current setting, they lead to very high condition numbers of the weight and stiffness matrices so that one cannot expect good results. The results in the first three columns of Table 2.2 illustrate that for $\ell \in \{4, 5, 6\}$ a stencil size $n > (\ell - 2)M$ is necessary. Therefore, we suggest to select the stencil size n for Cartesian nodes for $\ell \geq 4$ via $n = (\ell - 1)M$.

ℓ	$n_{\min}$	M	$2M$	$2M + \lfloor \ln(2M) \rfloor$
2	16	10	20	22
3	35	20	40	43
4	98	35	70	74
5	202	56	112	116
6	> 359	84	168	173
7	> 512	120	240	245
8	> 704	165	330	335

Table 2.2: Required and recommended stencil sizes for PHS on a three-dimensional Cartesian grid.

As discussed in Subsection 2.3.4, using a stencil size of approximately $n \approx 2M$ seems reasonable for Cartesian nodes with $\ell \leq 3$ or unstructured nodes. Further tests on unstructured grids (not included here) have shown an only slight advantage when using $n = 2M + \lfloor \ln(2M) \rfloor$ compared to $n = 2M$ (their difference in the stencil size is negligible as illustrated in the last two columns of Table 2.2) for the error (2.27) and the condition numbers of the weight and stiffness matrices, especially in the case of larger degrees of

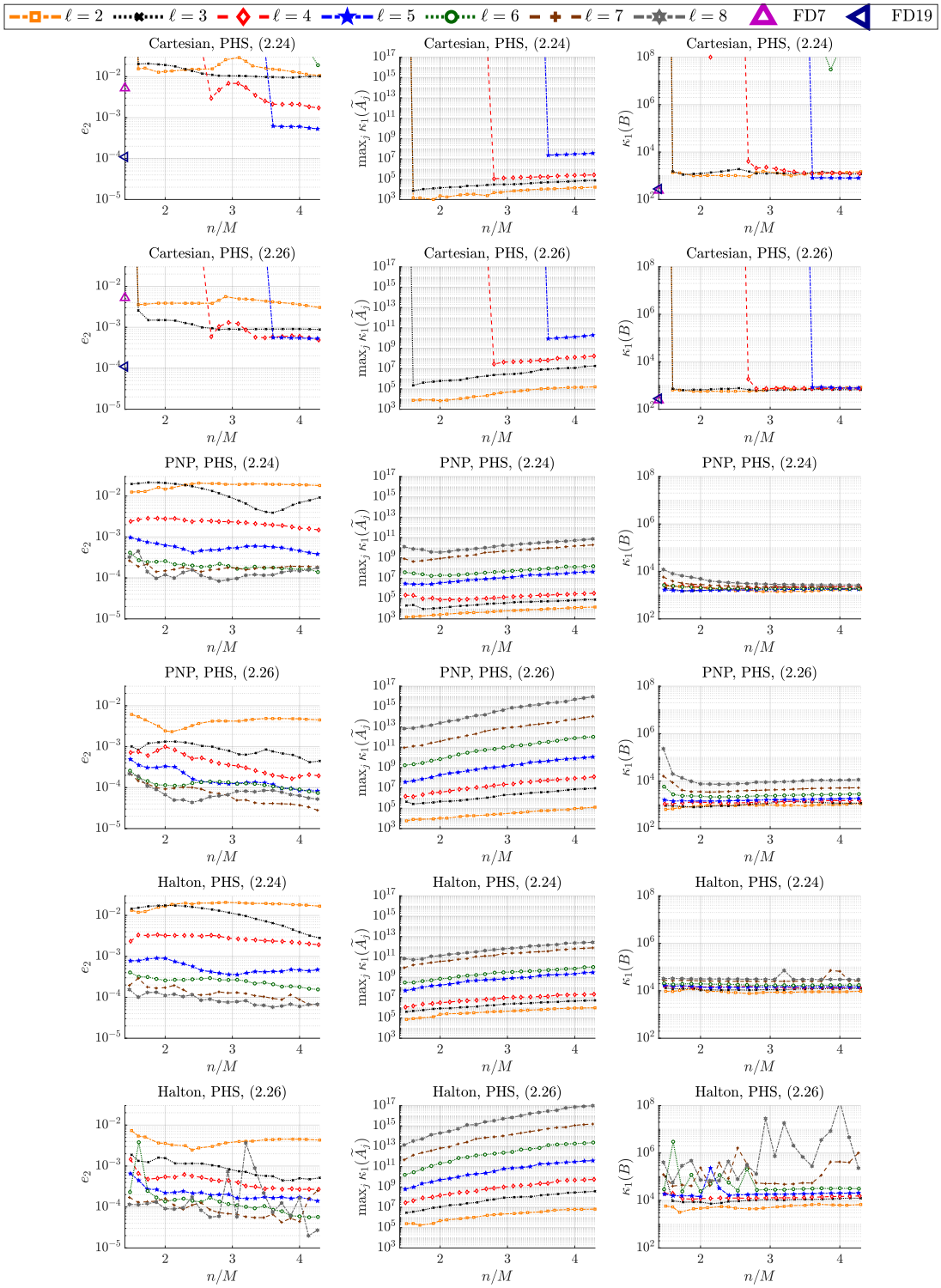

Figure 2.7: Errors (2.27) and condition numbers as a function of the quotient of stencil size n and the number of augmented polynomials M (2.6) for the PHS(k) generating function with $k \in \{2, 3, 4\}$ given by (2.24) (odd rows) and $k \in \{3, 4, 5, 6, 7, 8\}$ given by (2.26) (even rows), degree of polynomial augmentation $\ell \in \{2, 3, 4, 5, 6, 7, 8\}$ and node distributions (Cartesian, PNP, Halton) with $N = 8000$ nodes and $N_I = 5832$ interior nodes.

polynomial augmentation (or on random nodes). Since the difference between $n = 2M$ and $n = 2M + \lfloor \ln(2M) \rfloor$ (or some similar stencil size) is typically rather small (as long as ℓ is sufficiently large (2.20)), we recommend (and will use ourselves in subsequent tests) the following stencil sizes,

$$n = \begin{cases} (\ell - 1)M, & \text{if Cartesian nodes are used and } \ell \geq 4, \\ 2M + \lfloor \ln(2M) \rfloor, & \text{otherwise.} \end{cases} \tag{2.29}$$

A larger stencil size not only increases the computational cost to compute the stencil weights but also leads to more non-zero entries in the stiffness matrix. We hence conclude that RBF-FD using PHS and polynomial augmentation with a large degree ℓ is not suitable/efficient for Cartesian nodes since the stencil size should increase with the degree ℓ of polynomial augmentation as for example suggested in (2.29). Compared to the search for a best shape parameter in infinitely smooth RBFs, the sensitivity of the results when moving away from the optimal setting is much reduced here.

The comparison between the plots in the odd and even rows demonstrates that larger PHS degrees (i. e., (2.26) instead of (2.24)) can decrease the errors but may eventually lead to numerical instabilities. The larger PHS degrees via (2.26) seem to be especially beneficial for smaller polynomial degrees $\ell \leq 4$ while they can lead to numerical instabilities for $\ell \geq 5$.

In Figure 2.8, we show results for the PHS(k) generating functions with stencil sizes given by (2.29), polynomial augmentation of degree $\ell = 3$ (for Cartesian, PNP and Halton nodes) with $k \in \{2, \ldots, 5\}$, $\ell = 5$ with $k \in \{2, \ldots, 7\}$ and $\ell \in \{6, 8\}$ with $k \in \{2, \ldots, 8\}$ (only for Halton nodes). In particular, in these tests k and ℓ are no longer connected via (2.24) or (2.26). However, the polynomial degree ℓ now determines the stencil size via (2.29). These tests show that violating the unisolvency condition (2.22) (by using (k, ℓ) combinations $(5, 3), (7, 5), (8, 6)$) can (especially for PNP and Halton nodes) significantly increase the errors (2.27) and the condition numbers of the weight and stiffness matrices. Furthermore, it can be observed that using higher values of k than those given by (2.24) can further decrease the error. The Equation (2.20) for the convergence order can be interpreted in the way that for increasing N and ℓ, the other parameters n and k become less influential. Our numerical tests indicate that for our settings (2.26) leads to smaller approximation errors compared to (2.24). However, [107] points out that using a larger k than given by (2.24) can lead to numerical instabilities (especially for a large degree ℓ).

The setups in Figure 2.9 are similar to the setup in the last row of Figure 2.8 (i. e., Halton nodes with $\ell = 8$). The first row shows results without scaling (i. e., Remark 2.7). This demonstrates that scaling can significantly reduce the condition numbers of the weight matrices (especially for larger N values) such that they are almost independent of N. However, the effect of scaling on the errors and the condition numbers of the stiffness matrices is negligible. The second row presents results without shift-and-scale (i. e., the stencil weights are computed via (2.11)). This indicates that shifting the stencil center to the origin can significantly reduce the condition numbers of the weight matrices

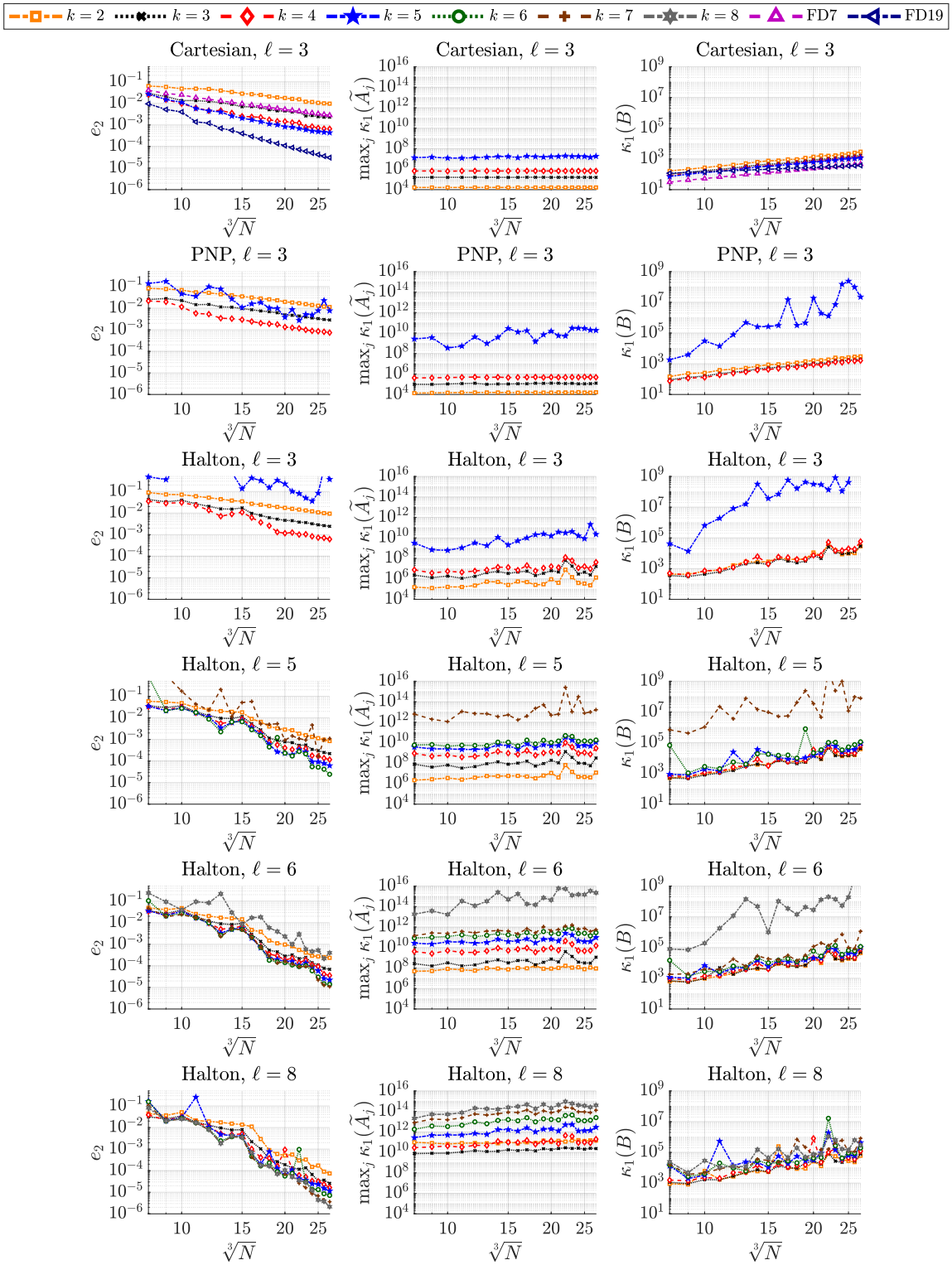

Figure 2.8: Errors (2.27) and condition numbers as a function of the number of nodes for the PHS(k) generating function r^{2k-1} with polynomial augmentation of degree $\ell = 3$ with $k \in \{2, \ldots, 5\}$, $\ell = 5$ with $k \in \{2, \ldots, 7\}$ and $\ell \in \{6, 8\}$ with $k \in \{2, \ldots, 8\}$ and stencil sizes n given by (2.29).

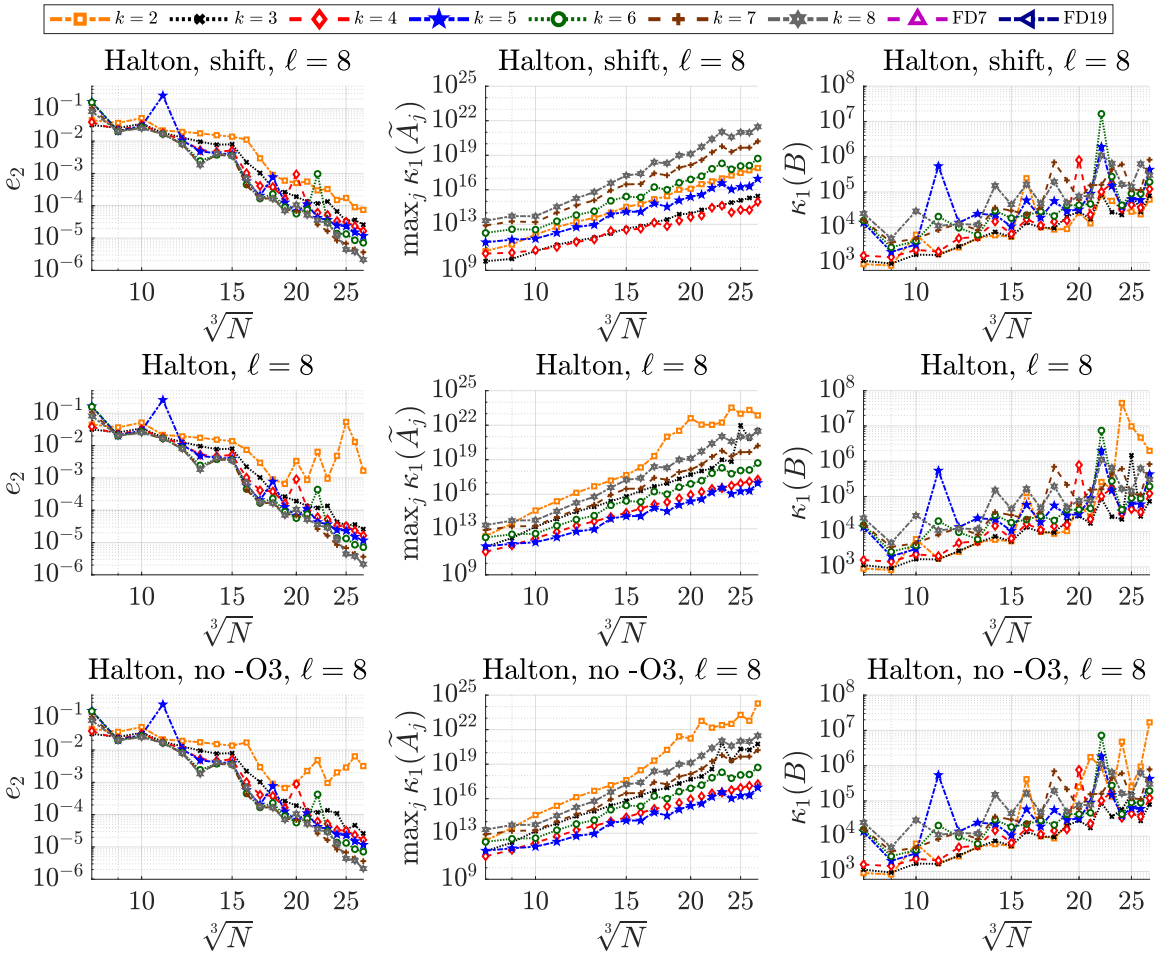

Figure 2.9: Errors (2.27) and condition numbers as a function of the number of Halton nodes for the PHS(k) generating function r^{2k-1} with polynomial augmentation of degree $\ell = 8$ with $k \in \{2, \dots, 8\}$ and stencil size n given by (2.29). Stencil weights via (2.11) and in first row with additional shift (i. e., Remark 2.7). Last row: without compiler flag *-O3*.

for the combinations of small PHS degrees (i. e., $k \in \{2, 3, 4\}$) with large degrees of polynomial augmentation (i. e., $\ell = 8$). Shifting can in addition significantly reduce the errors and the condition numbers of the stiffness matrices (see $k = 2$). The last row is identical to the second row except that the level of compiler optimization was changed (i. e., compilation without *-O3*) and shows that even the level of compiler optimization can have a visible effect on the errors and the condition numbers (see $k = 2$). This illustrates (as mentioned in [107]) that also a too small k (i. e., $k < k_{\min}$ (2.24)) can lead to numerical instabilities.

In Figure 2.10, we show results w. r. t. the total number of nodes N for the generating function PHS(k). Each color/marker represents a different degree of polynomial augmentation $\ell \in \{2, 3, 4, 5, 6, 7, 8\}$. This degree ℓ determines k by (2.26) and n by (2.29).

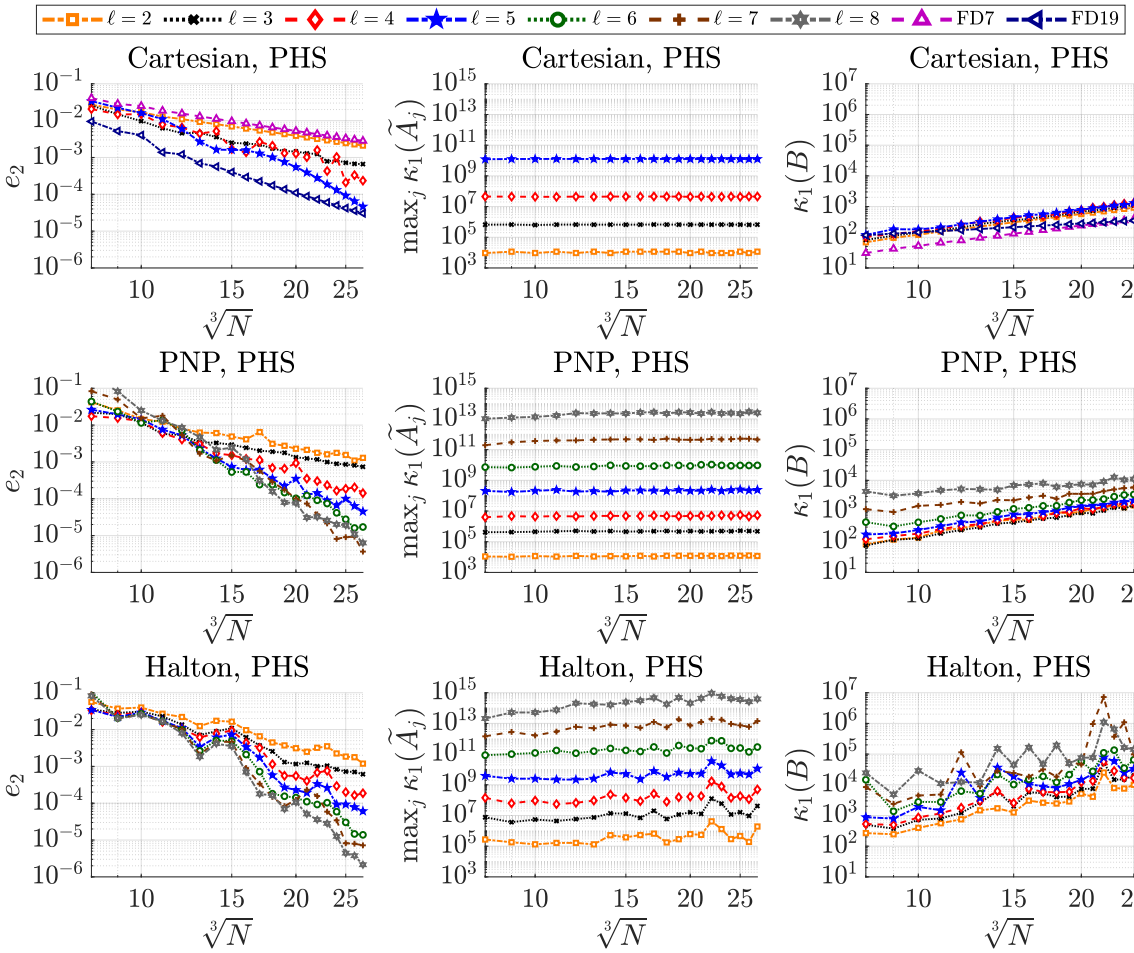

Figure 2.10: Errors (2.27) and condition numbers as a function of the number of nodes for PHS(k) generating function, degrees of polynomial augmentation $\ell \in \{2, 3, 4, 5, 6, 7, 8\}$ (k and stencil size n are determined by (2.26) and (2.29), resp., for a fixed ℓ).

Further tests (not included here) on unstructured nodes with stencil sizes n determined via $n \in \{2M, 3M\}$ instead of (2.29) do not lead to significant changes in the errors (2.27) or the condition numbers of the weight or stiffness matrices whereas $n = M + \lfloor \ln(2M) \rfloor$ turned out to be too small (in most cases), leading to significantly larger errors and condition numbers of the weight and stiffness matrices.

For Cartesian nodes, we show results only for $\ell \leq 5$ since larger degrees require very large stencil sizes, rendering the RBF-FD approach computationally very costly and hence not competitive (e. g., $n = 420$ for $\ell = 6$ on a Cartesian grid versus $n = 335$ for $\ell = 8$ on a PNP or Halton grid).

We find that for $\ell \in \{2, 3, 4, 5\}$, we obtain comparable errors on Cartesian, PNP and Halton node sets (for significantly larger stencil sizes on Cartesian grids for $\ell \geq 4$ (2.29)). Larger degrees of polynomial augmentation (and with it larger stencils) lead to smaller

discretization errors but larger condition numbers in the weight matrices.

The tests in this subsection illustrated that the PHS(k) generating function can be used successfully as long as the parameter k, the degree ℓ of polynomial augmentation and the stencil size n are adjusted correspondingly.

2.4.4 Hybrid (HYB) RBFs and comparison with other RBFs

In this subsection, we show numerical results for hybrid generating functions HYB(γ) and comparisons to GA, IMQ, IQ, MQ and PHS (see Table 2.1). We will mostly use $\gamma = 10^{-5}$ (which fulfills (2.18)) but also include some numerical experiments with $\gamma \in \{10^{-9}, 10^{-1}\}$ to illustrate the influence of the γ value and the consequences of violating (2.18). The results of this subsection are presented through the following figures.

- Figure 2.11 (as Figure 2.4, with HYB(10^{-5}) instead of GA): Influence of the shape parameter ε on different grid types (Cartesian, PNP, Halton) for different stencil sizes n, fixed problem size N.
- Figure 2.12 (as Figure 2.5, with HYB(10^{-5}) instead of GA): Influence of the stencil size n on different grid types (Cartesian, PNP, Halton) for different shape parameters ε, fixed problem size N, polynomial augmentation of degree $\ell = 0$.
- Figure 2.13 (as Figure 2.6, with HYB(10^{-5}) instead of GA): Influence of the problem size N on different grid types (Cartesian, PNP, Halton) for different stencil sizes n, a problem size dependent shape parameter ε (2.28), polynomial augmentation of degree $\ell = 0$.
- Figure 2.14: Influence of the problem size N on different grid types (Cartesian, PNP, Halton, random) for different generating functions (GA, IMQ, IQ, MQ, HYB(10^{-9}), HYB(10^{-5}), HYB(10^{-1})), a problem size dependent shape parameter ε (2.28), stencil size $n \in \{50, 131\}$, polynomial augmentation of degree $\ell = 0$.
- Figure 2.15: Influence of the shape parameter ε for HYB(10^{-5}) on different grid types (Cartesian, PNP, Halton) for different degrees of polynomial augmentation, stencil size $n = 90$, fixed problem size N.
- Figure 2.16: Influence of shift (Remark 2.7) for HYB(10^{-9}), HYB(10^{-5}), HYB(10^{-1}) on Halton nodes for stencil size $n = 335$ and polynomial augmentation of degree $\ell = 8$.
- Figure 2.17: Influence of the problem size N on different grid types (Cartesian, PNP, Halton, random) for HYB(10^{-9}), HYB(10^{-5}), HYB(10^{-1}) (with a problem size dependent shape parameter ε (2.28)) and PHS(k) with k given by (2.26), stencil size n given by (2.29), degree of polynomial augmentation $\ell \in \{3, 5, 8\}$.

The results of Figure 2.11 may be compared to those in Figure 2.4 since we have only exchanged the Gaussian generating function and now use HYB(10^{-5}). A noticeable difference can be observed as the shape parameter ε goes to zero: the condition numbers of weight and stiffness matrices now remain bounded, and while there still is an optimal

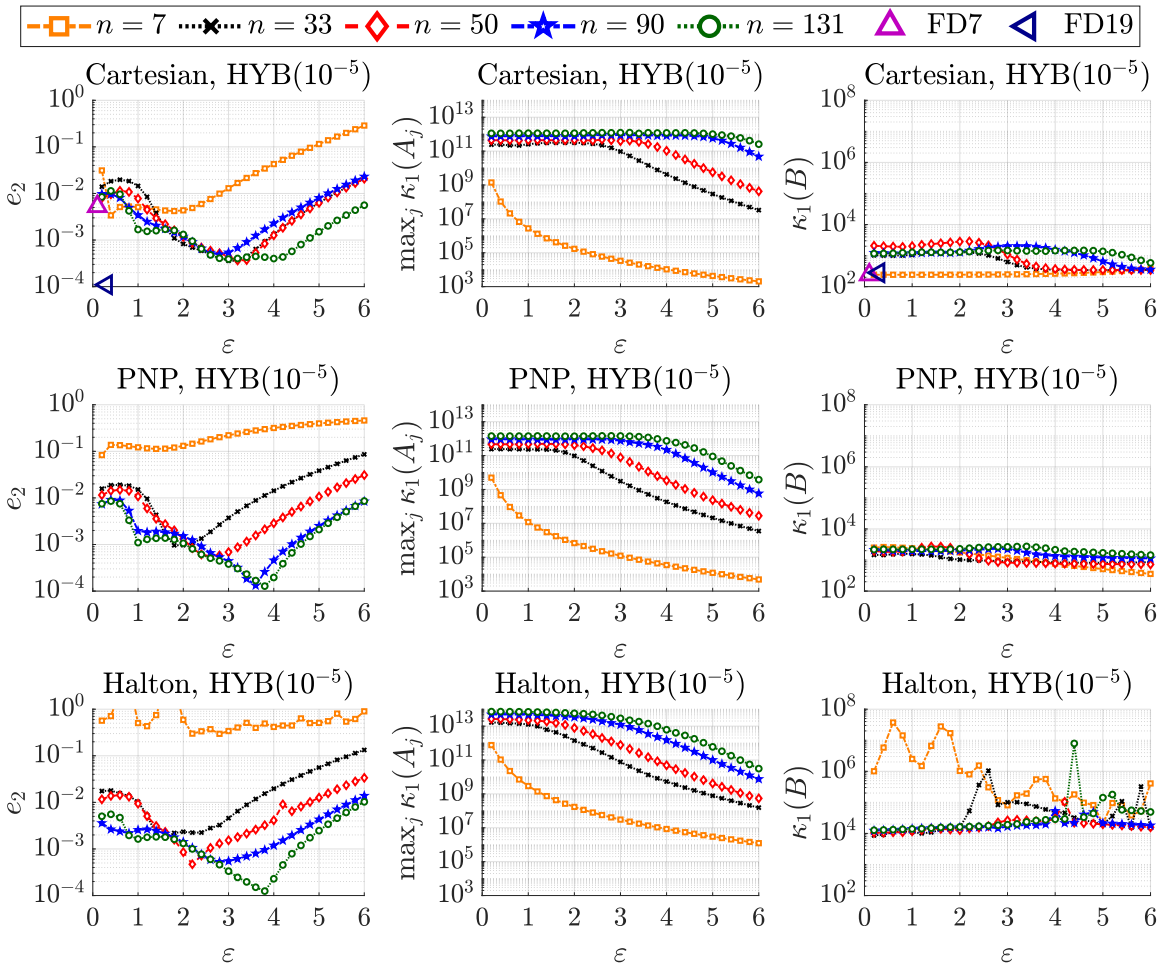

Figure 2.11: Errors (2.27) and condition numbers as a function of the shape parameter ε for the HYB(10^{-5}) generating function, stencil sizes $n \in \{7, 35, 50, 90, 131\}$ and node distributions (Cartesian, PNP, Halton) with $N = 8000$ nodes and $N_I = 5832$ interior nodes.

shape parameter $\varepsilon \in (1, 4)$, the errors/condition numbers no longer blow up for $\varepsilon \to 0$.

Figure 2.12 compares to Figure 2.5 in the same sense that only the Gaussian generating function has been replaced by HYB(10^{-5}) (and, again, only every fifth marker is visible). On the Cartesian grid, we observe the same jump in accuracy at the stencil size $n = 33$. When increasing the stencil size n, the condition numbers now remain bounded (and significantly smaller than for the Gaussian generating function) so that the discretization errors no longer blow up due to ill-conditioning for large n.

Figure 2.13 shows test results for the same setting as in Figure 2.6, again for HYB(10^{-5}) instead of GA. It shows that HYB(10^{-5}) can lead to similar errors as the GA generating function with the additional benefit of numerical stability for smaller shape parameters, larger stencil sizes and finer discretizations.

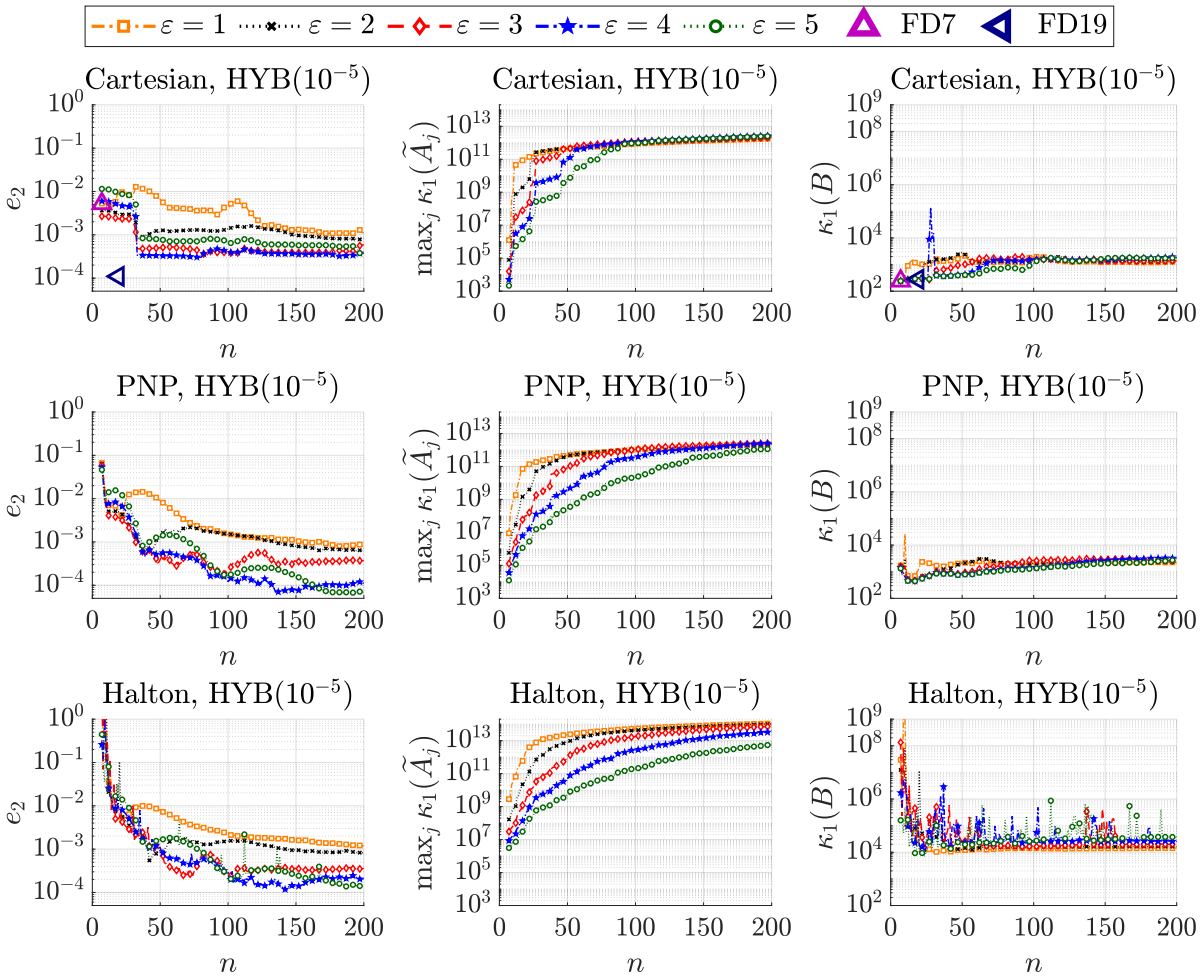

Figure 2.12: Errors (2.27) and condition numbers as a function of the stencil size n for the HYB(10^{-5}) generating function, shape parameters $\varepsilon \in \{1, 2, 3, 4, 5\}$, polynomial augmentation of degree $\ell = 0$ and node distributions (Cartesian, PNP, Halton) with $N = 8000$ nodes and $N_I = 5832$ interior nodes.

In Figure 2.14, we use stencil sizes $n = 50$ (top four rows) and $n = 131$ (bottom two rows) to compare results for the generating functions GA, IMQ, IQ, MQ, HYB(10^{-9}), HYB(10^{-5}), HYB(10^{-1}) on Cartesian, PNP, Halton and random nodes. For each combination of generating function, grid type and stencil size, we beforehand numerically determined the constant $c \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$ in (2.28) leading to the smallest approximation errors over a wide range of tested problem sizes N, see Table 2.3.

We observe that on all except the random node sets, the RBF-FD variants lead to more accurate approximations than the 7-point FD stencil (except for very small problem sizes N). On PNP and Halton grids, all except HYB(10^{-1}) lead to approximation errors comparable or even better than the 19-point FD stencil on the Cartesian grid.

Comparing the three hybrid versions, we note that HYB(10^{-1}) (compared to HYB(10^{-5})) reduces the condition numbers but increases the error, while HYB(10^{-9}) leads to con-

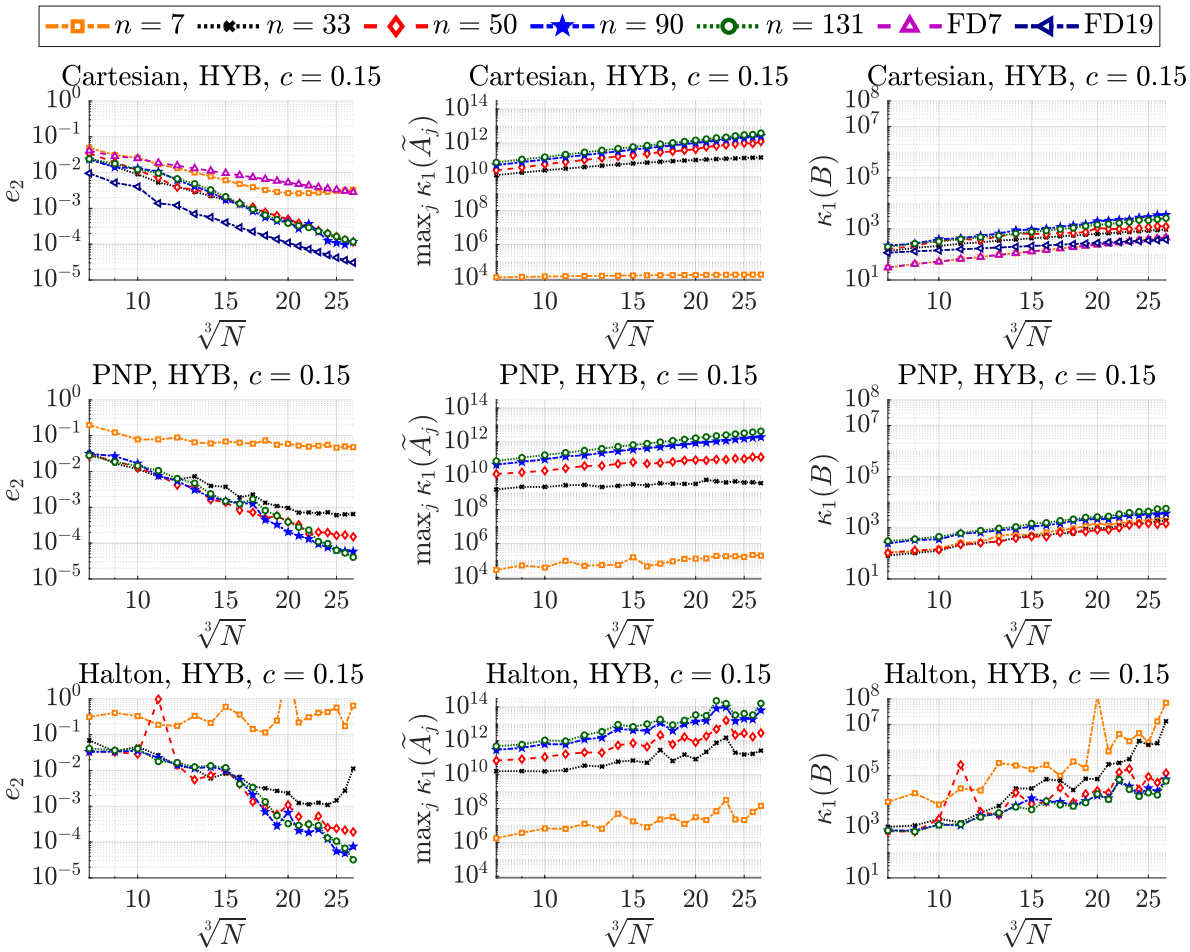

Figure 2.13: Errors (2.27) and condition numbers as a function of the number of nodes for the HYB(10^{-5}) generating function with polynomial augmentation of degree $\ell = 0$, $c = 0.15$ (2.28) and stencil sizes $n \in \{7, 33, 50, 90, 131\}$.

dition numbers and approximation errors comparable to those for the infinitely smooth RBFs.

For $n = 131$, we include only results for PNP and Halton nodes since there is typically no (or only a slight) improvement on Cartesian nodes compared to $n = 50$ (or, in fact, $n = 33$), and the results for random nodes are qualitatively similar to those for Halton nodes. Increasing from $n = 50$ to $n = 131$ on PNP or Halton nodes leads to a significant reduction in the errors (accompanied by an increase in the condition numbers of the weight matrices) for all generating functions except for HYB(10^{-1}). The behavior for HYB(10^{-1}) is similar than for PHS, i. e., the errors are mainly decreased by increasing the polynomial degree ℓ and not the stencil size n.

In Figure 2.15, we add polynomial augmentation to the HYB(10^{-5}) generating function and use shifting in the last row (i. e., stencil weights via Remark 2.7). We also add two horizontal (dashed gray/green) lines for the (shape parameter independent) PHS(2)

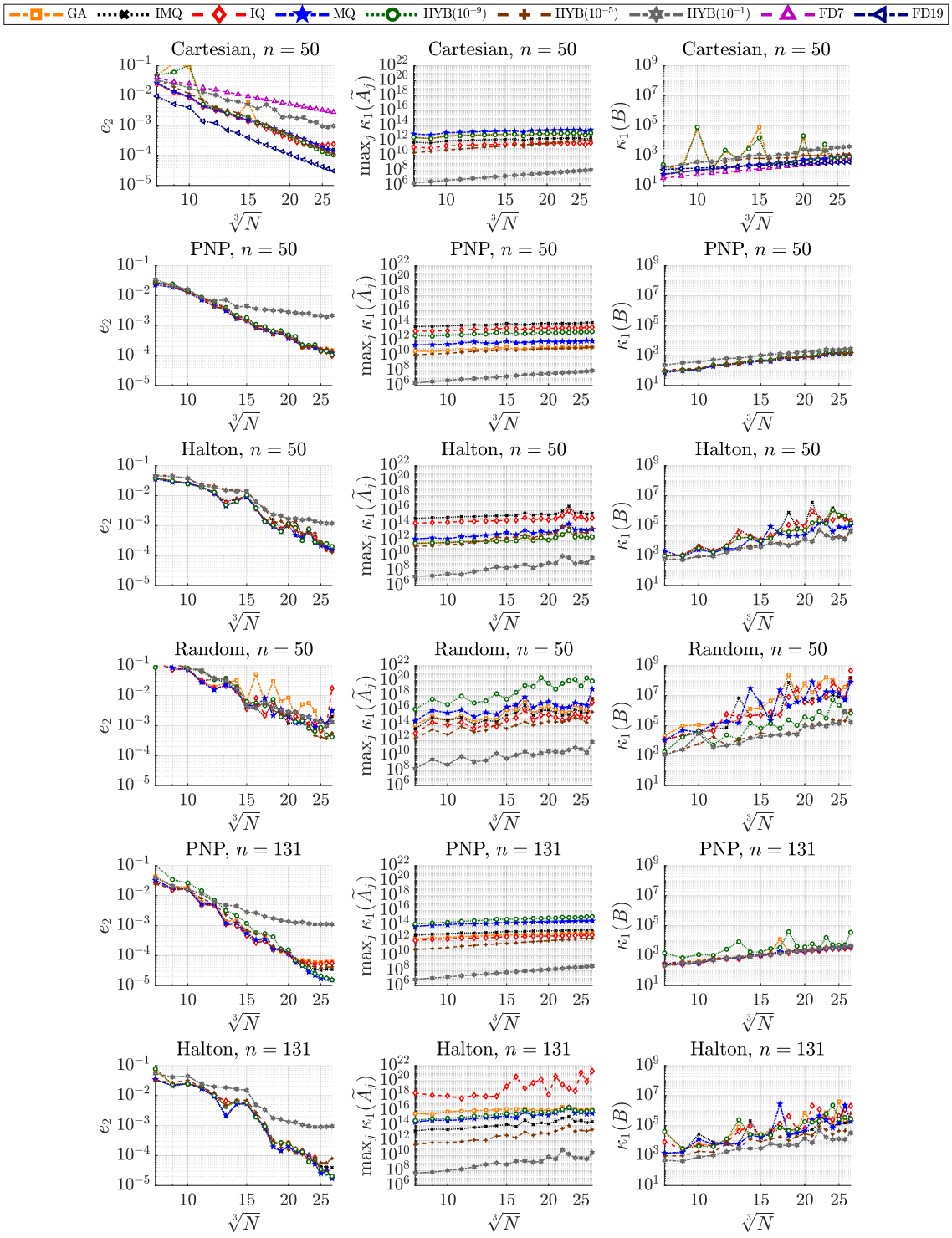

Figure 2.14: Errors (2.27) and condition numbers as a function of the number of nodes for several generating functions with shape parameter $\varepsilon = c\sqrt[3]{N}$, c as in Table 2.3, polynomial augmentation of degree $\ell = 0$, stencil size $n = 50$ for Cartesian, PNP, Halton and random nodes and stencil size $n = 131$ for PNP and Halton nodes.

grid type	Cartesian	PMP		Halton		random
stencil size	$n = 50$	$n = 50$	$n = 131$	$n = 50$	$n = 131$	$n = 50$
GA	0.15	0.15	0.2	0.15	0.15	0.15
IMQ	0.1	0.05	0.1	0.05	0.1	0.1
IQ	0.1	0.05	0.1	0.05	0.05	0.1
MQ	0.1	0.1	0.1	0.1	0.1	0.1
HYB(10^{-9})	0.1	0.1	0.15	0.15	0.15	0.05
HYB(10^{-5})	0.1	0.15	0.2	0.1	0.2	0.1
HYB(10^{-1})	0.2	0.25	0.15	0.25	0.15	0.25

Table 2.3: Summary of constants c in (2.28) for tests in Figure 2.14.

(with mainly shifted and scaled stencils according to Subsection 2.4.3, just shifted stencils are only used in the last row) results using the two polynomial augmentation degrees $\ell \in \{3, 5\}$, resp. (and otherwise identical setting). On the Cartesian grid, HYB(10^{-5}) with $\ell = 5$ failed for $\varepsilon \in \{0.5, 1\}$ (with a breakdown in the LU factorization of the weight matrices) which we indicate by the dummy values 10^{39} and 10^{22} for the condition numbers of the weight and stiffness matrices, resp. We recall that a minimal stencil size n depending on the polynomial augmentation degree ℓ is necessary for the weight matrices to be non-singular. For $\ell = 6$, the polynomial space Π_6 (2.5) for $d = 3$ has dimension 84 (2.6), hence the fixed stencil size $n = 90$ satisfies this constraint for all degrees of polynomial augmentation $\ell \in \{-1, 0, \dots, 6\}$. Here, $\ell = -1$ denotes the case of no polynomial augmentation.

The best results in terms of the errors (2.27) and condition numbers of the weight and stiffness matrices are obtained for Cartesian nodes for $\ell = 2$ and for PNP as well as Halton nodes for $\ell = 4$. Larger degrees ℓ do not appear advisable since they increase the computational cost without reducing the approximation error. The magnitude of the optimal shape parameter increases (and the range of suitable shape parameters widens) as the degree of polynomial augmentation increases (especially on the PNP and Halton grids), and an appropriately chosen shape parameter leads to significantly better results than PHS(2) by itself. While $\varepsilon \to 0$ leads to blow-up (and no improvement though polynomial augmentation) for the Gaussian kernel, HYB(γ) with polynomial augmentation behaves similar to PHS with polynomial augmentation as $\varepsilon \to 0$ (since the Gaussian approximates a constant contained in the polynomial augmentation).

Shifting the stencil to the origin (without scaling in the case of PHS) has no visible influence on the errors (2.27) except for the numerical unstable case with degree $\ell = 6$. This case with $\ell = 6$ illustrates the propagation of errors (see Subsection 2.2.3), i. e., the larger the condition numbers of the weight and the stiffness matrices, the larger is the influence of rounding errors. Furthermore, shifting the stencil to the origin can significantly change (i. e., increase as well as decrease) the condition numbers of the weight and stiffness matrices for the setup with $\ell = 6$. For the degrees $\ell \in \{3, 4, 5\}$, the condition numbers of the weight matrices for HYB with a larger shape parameter ε are significantly reduced by shifting the stencil, whereas all other cases with HYB do not lead

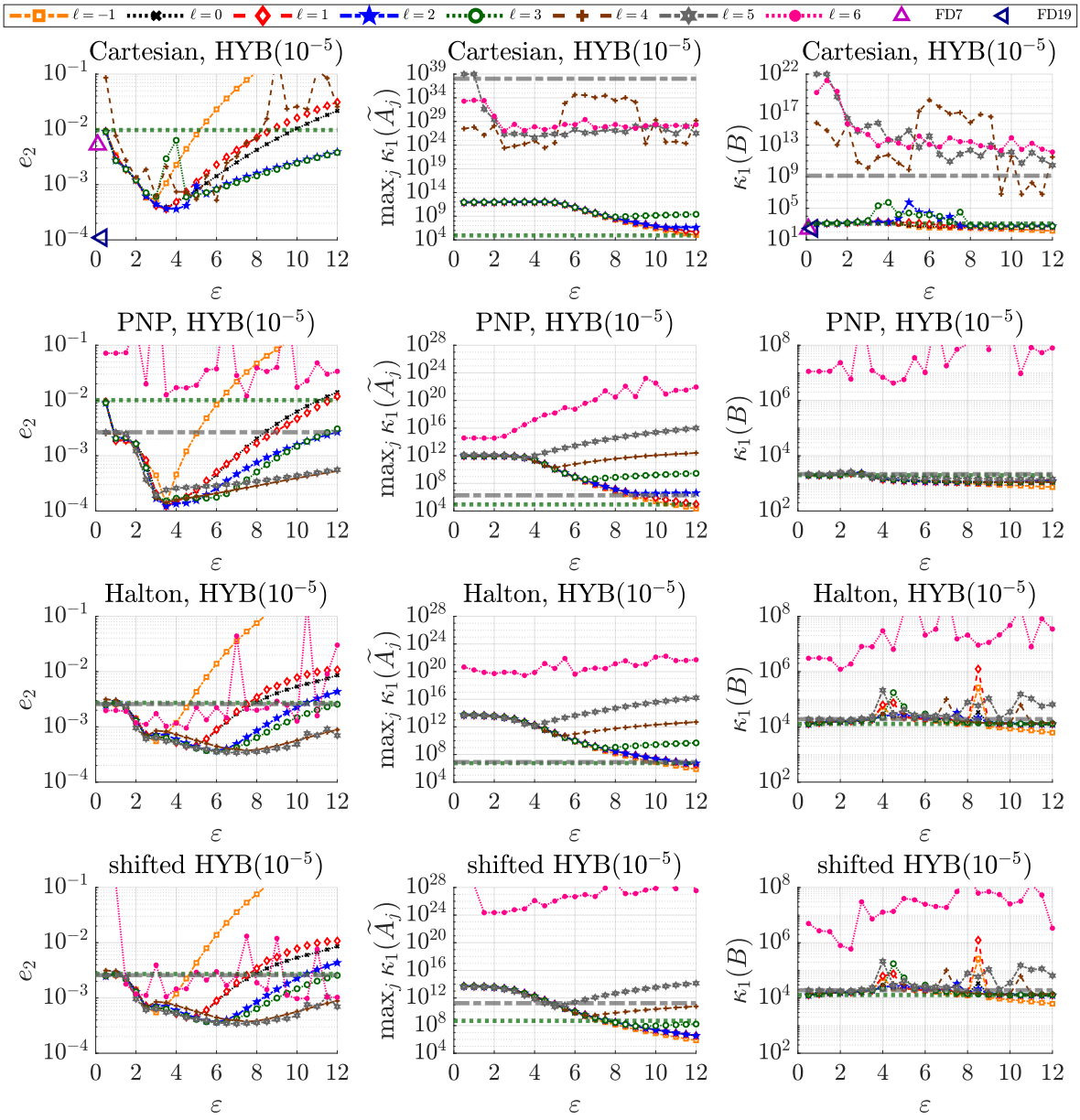

Figure 2.15: Errors (2.27) and condition numbers as a function of the shape parameter ε for the HYB(10^{-5}) generating function, stencil size $n = 90$, polynomial augmentation of degree $\ell \in \{-1, 0, \ldots, 6\}$ and node distributions (Cartesian, PNP, Halton) with $N = 8000$ nodes and $N_I = 5832$ interior nodes. (Horizontal gray/green dashed lines show results for ε-independent PHS(2) with $\ell \in \{3, 5\}$.) Last row identical to last but one row, except that HYB is shifted (i. e., Remark 2.7).

to a visible change in the condition numbers of the weight or stiffness matrices. However, the condition numbers of the weight matrices for PHS are significantly increased for the shifted version (similarly to Figure 2.9) since no scaling of the stencils is used.

This figure as well as many additional tests (not included here) with a setting as in Figure 2.7 but for HYB(γ) for $\gamma \in \{10^{-9}, 10^{-5}, 10^{-1}\}$ have shown that some results for PHS regarding polynomial augmentation carry over to HYB (e. g., Cartesian nodes need larger stencil sizes than unstructured nodes) and shifting the stencil to the origin usually, i. e., in setups without numerical instabilities, e. g., here for $\ell \leq 5$, leads to almost identical results). Hence, for the remaining tests, we use (2.29) to determine the stencil size n for HYB when used in combination with larger degrees ℓ.

Figure 2.16 shows results on Halton nodes with polynomial degree $\ell = 8$ for several generating functions (HYB(10^{-9}), HYB(10^{-5}), HYB(10^{-1})) with shape parameter $\varepsilon = c\sqrt[3]{N}$ (2.28). Each color/marker represents a different constant $c \in \{0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$. These c constants include the in Table 2.3 for Halton nodes with stencil size $n = 131$ proposed c values. Larger c constants are tested in addition since Figure 2.15 indicates that larger degrees ℓ typically lead to larger optimal shape parameters ε. The top three rows use the standard approach (2.11) to compute the stencil weights whereas the last three rows utilize shifting of the stencil centers to the origin (i. e., computing the stencil weights via Remark 2.7). This illustrates that shifting can significantly reduce the errors (2.27) and the condition numbers of the weight and stiffness matrices when larger degree polynomials are augmented. Furthermore, the constants c leading to the smallest approximation errors over a wide range of tested problem sizes N for the last three rows are summarized in the sixth column (i. e., Halton with $\ell = 8$) of Table 2.4.

The setup in Figure 2.17 is similar to Figure 2.14 with PHS instead of the generating functions GA, IMQ, IQ, MQ, and now larger degrees ℓ. Each color/marker represents a different generating function (HYB(10^{-9}), HYB(10^{-5}), HYB(10^{-1}) and PHS(k)). The top four rows of plots show results for $\ell = 3$ on Cartesian, PNP, Halton and random nodes, resp. The polynomial degree determines k and n via (2.29) and (2.26), leading to $(\ell, n, k) = (3, 43, 4)$. In the bottom two rows, we show results for the two parameter combinations $(\ell, n, k) \in \{(5, 116, 5), (8, 335, 8)\}$. We use shifting with the HYB generating functions solely in the last row for $\ell = 8$ since we observed in Figures 2.7 and 2.16 that shifting is usually beneficial if the degree ℓ is large (whereas the results are almost identical for smaller degrees). For each combination of γ value in HYB(γ), grid type and degree of polynomial augmentation, we beforehand numerically determined the constant $c \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$ for the problem size dependent shape parameter in (2.28) leading to the smallest approximation errors over a wide range of tested problem sizes N, see Table 2.4. The PHS generating function needs polynomial

grid type	Cartesian	PNP	Halton			random
degree	$\ell = 3$	$\ell = 3$	$\ell = 3$	$\ell = 5$	$\ell = 8$	$\ell = 3$
HYB(10^{-9})	0.25	0.15	0.2	0.25	0.2	0.35
HYB(10^{-5})	0.1	0.1	0.1	0.2	0.25	0.1
HYB(10^{-1})	0.3	0.3	0.25	0.35	0.4	0.25

Table 2.4: Summary of constants c in (2.28) for tests in Figure 2.17.

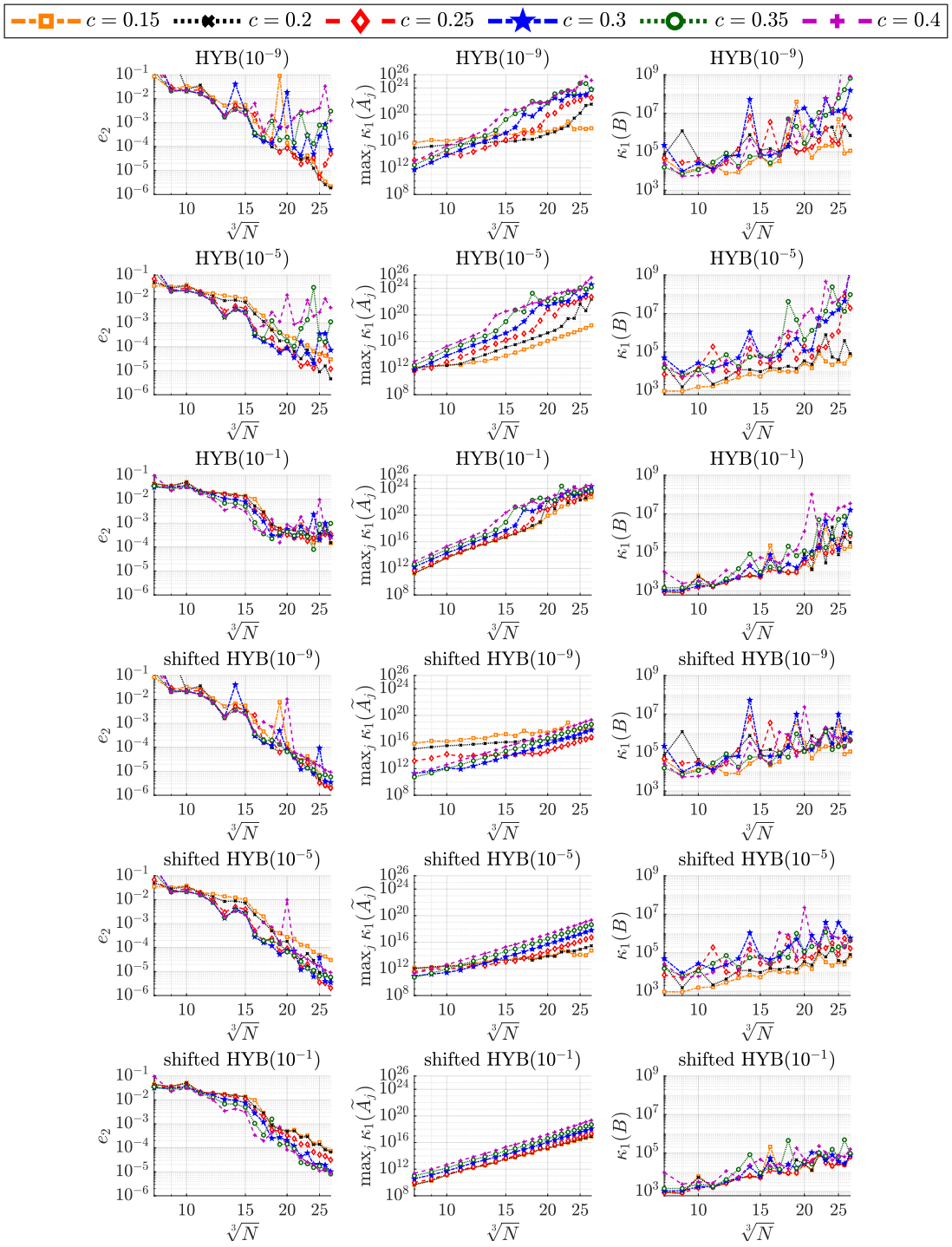

Figure 2.16: Errors (2.27) and condition numbers as a function of the number of nodes for several generating functions (HYB(10^{-9}), HYB(10^{-5}), HYB(10^{-1})) with shape parameter $\varepsilon = c\sqrt[3]{N}$ with constants $c \in \{0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$ on Halton nodes, degree $\ell = 8$ and stencil size n given by (2.29). First three rows without shifting (i. e., (2.11)) and last three rows with shifting (i. e., Remark 2.7).

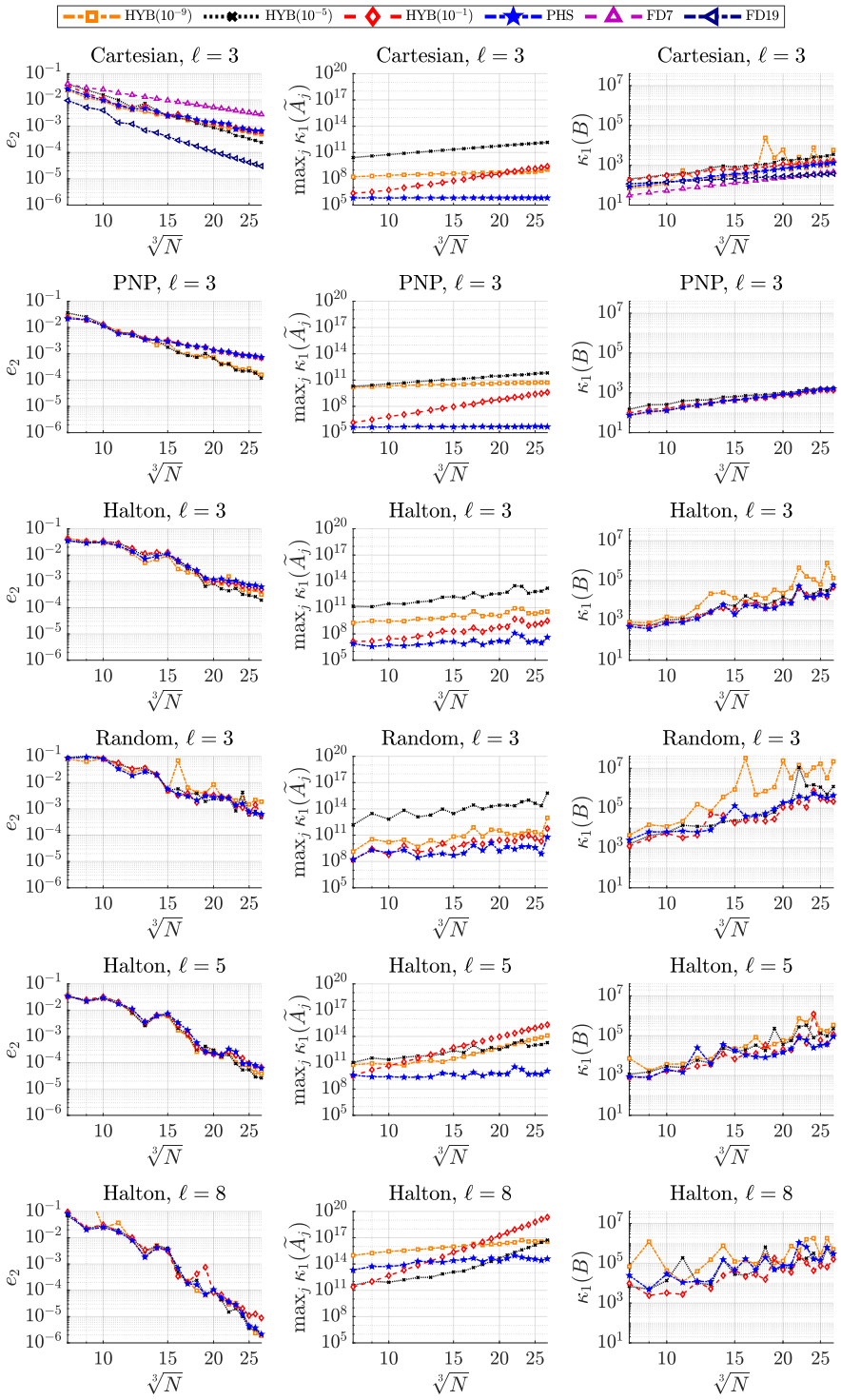

Figure 2.17: Errors (2.27) and condition numbers as a function of the number of nodes for several generating functions with fixed $(\ell, n, k) = (3, 43, 4)$ for Cartesian, PNP, Halton and random nodes (top four rows) and $(\ell, n, k) \in \{(5, 116, 5), (8, 335, 8)\}$ (i. e., given ℓ, then n and k follow by (2.29) and (2.26)) and shape parameter $\varepsilon = c\sqrt[3]{N}$, c as in Table 2.4, for the HYB generating functions on Halton nodes (bottom two rows).

augmentation of degree $\ell \geq D$ (2.20) in order to converge and we use respective degrees for the HYB generating function as well. The γ value seems to be mainly important for the numerical stability and not so much for the errors in the case of larger degrees ℓ. Tests with $\ell = 5$ on Cartesian nodes (not included here) showed that Cartesian nodes are not an advisable option for RBF-FD with HYB in combination with larger degrees of polynomial augmentation ℓ since the required stencil sizes are significantly larger than for unstructured nodes without leading to smaller errors, similar to the PHS case studied in Subsection 2.4.3. Tests for $\ell \in \{5, 8\}$ with k according to (2.26) show a decrease in the approximation errors as ℓ increases with hardly any noticible difference between HYB(10^{-9}), HYB(10^{-5}) and PHS whereas HYB(10^{-1}) can lead to slightly higher errors.

2.4.5 Different geometry with a different node distribution

Here, we add additional tests where the domain is a sphere $\Omega = \{x \in \mathbb{R}^3 \colon \|x\|_2 < 1\}$ instead of the cube $\Omega = (0, 1)^3$. The nodes are created in two steps as in [30]: First the boundary nodes are created by the PA algorithm [31], a recently introduced node placing algorithm for arbitrary surfaces that works in the parametric space and needs only a parametrization of the surface as well as its Jacobian. Then these nodes are used as start nodes in the PNP algorithm [115] to create the interior nodes. Both algorithms run with the same constant spacing function $h \equiv (\nu)^{-1}$ with $\nu \in \{8, \cdots, 19\}$. In this subsection, we provide the following numerical results (with slightly changed x-axis since the numbers of nodes N and interior nodes N_I are different for the cube and the sphere) to illustrate the influence of the domain Ω.

- Figure 2.18 (top and bottom row as Figures 2.6 and 2.13, resp., on a sphere (and different node distribution) instead of a cube): Influence of the problem size N for different stencil sizes n, a problem size dependent shape parameter ε (2.28), polynomial augmentation of degree $\ell = 0$.
- Figure 2.19 (as Figure 2.10 on a sphere (and different node distribution) instead of a cube): Influence of the problem size N for different degrees ℓ of polynomial augmentation which in turn determine k in PHS(k) and the stencil size n via (2.26) and (2.29), resp.

Figure 2.18 illustrates that the behavior on the sphere (with adjusted c value) is essentially the same as on the cube: GA (top row) and HYB(10^{-5}) (bottom row) generating functions lead to similar accuracies for (approximately) optimal shape parameters as long as γ is not too large. The value of the optimal parameter c in (2.28) that determines the shape parameter ε needs to be slightly smaller than for the cube, namely $c = 0.1$ instead of $c = 0.15$.

Figure 2.19 shows that no fine-tuning of the involved parameters is necessary to apply the (shape parameter independent) PHS setup on a different geometry. However, (similar as in Figures 2.15 and 2.17) the PHS generating function leads to lower accuracies than the GA or HYB generating functions (with finetuned shape parameters). Furthermore, the GA or HYB generating functions can reach these higher accuracies with lower

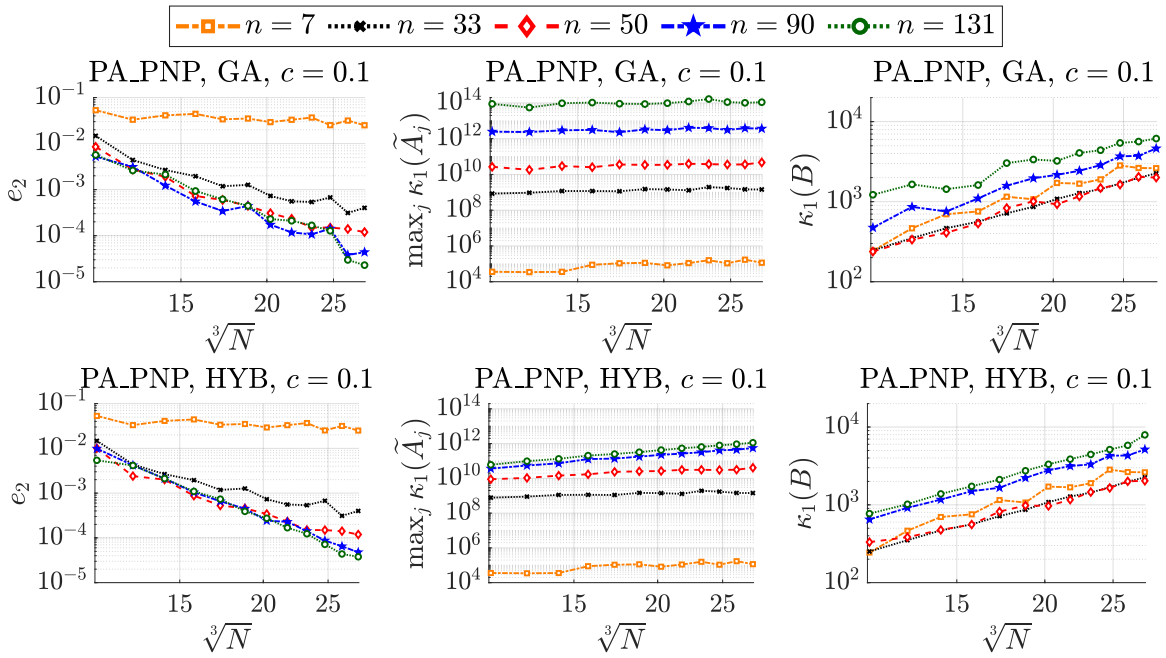

Figure 2.18: Errors (2.27) and condition numbers as a function of the number of nodes for the GA (top row) and HYB(10^{-5}) (bottom row) generating functions with polynomial augmentation of degree $\ell = 0$, $c = 0.1$ (2.28) and stencil sizes $n \in \{7, 33, 50, 90, 131\}$.

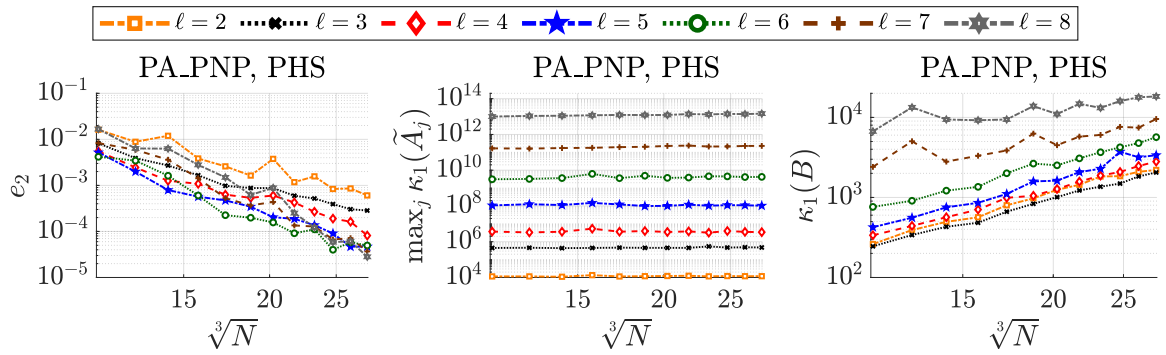

Figure 2.19: Errors (2.27) and condition numbers as a function of the number of nodes for PHS(k) generating function, degrees of polynomial augmentation $\ell \in \{2, 3, 4, 5, 6, 7, 8\}$ (k and stencil size n are determined by (2.26) and (2.29), resp., for a fixed ℓ).

computational cost, if they are used with a smaller degree of polynomial augmentation ℓ.

Further tests for the sphere are not included here since the observations are qualitatively comparable to the observations for the cube domain.

2.4.6 Convection diffusion equation

The following figures show additional tests, where the differential operator $\mathcal{L} = -\nu\Delta + b^T\nabla$ with $\nu \in \{10^{-j} : j \in \{0,1,2,3,4,5\}\}$ and b as the convective flow from problem 3D1 in [90], namely $b\colon \mathbb{R}^3 \to \mathbb{R}^3$ with

$$b(x) = \begin{pmatrix} 2x_1(1-x_1)(2x_2-1)x_3 \\ -(2x_1-1)x_2(1-x_2) \\ -(2x_1-1)(2x_2-1)x_3(1-x_3) \end{pmatrix},$$

is used instead of $\mathcal{L} = -\Delta$. The analytical solution of the partial differential Equation (2.12a), (2.12b) is again the three-dimensional version of Franke's function F (2.25), i. e., the right hand side function f in (2.12a) is changed from

$$-\Delta F \qquad \text{to} \qquad -\nu\Delta F + b^T\nabla F.$$

We focus here on the PHS and the HYB generating functions (since HYB(γ) leads to similar results as GA as long as γ is sufficiently small). The results of this subsection are presented through the following figures.

- Figure 2.20 (as errors (2.27) in Figure 2.10 for the convection-diffusion problem instead of the Laplace problem): Influence of the problem size N for different degrees ℓ of polynomial augmentation which in turn determine k in PHS(k) and the stencil size n via (2.26) and (2.29), resp.
- Figure 2.21 (as condition of the stiffness matrices in Figure 2.10 for the convection-diffusion problem instead of the Laplace problem): Influence of the problem size N for different degrees ℓ of polynomial augmentation which in turn determine k in PHS(k) and the stencil size n via (2.26) and (2.29), resp.
- Figure 2.22: Influence of the problem size N for different degrees ℓ of polynomial augmentation which in turn determine k in PHS(k) and the stencil size n via (2.24) and (2.29), resp.
- Figure 2.23 (as errors (2.27) in Figure 2.11 for the convection-diffusion problem instead of the Laplace problem): Influence of the shape parameter ε for different stencil sizes n, fixed problem size N.
- Figure 2.24 (as errors (2.27) in Figure 2.13 for the convection-diffusion problem instead of the Laplace problem): Influence of the problem size N for different stencil sizes n, a problem size dependent shape parameter ε (2.28), polynomial augmentation of degree $\ell = 0$.
- Figure 2.25 (as errors (2.27) in Figure 2.15 for the convection-diffusion problem instead of the Laplace problem): Influence of the shape parameter ε for HYB(10^{-5}) for different degrees of polynomial augmentation, stencil size $n = 90$, fixed problem size N.

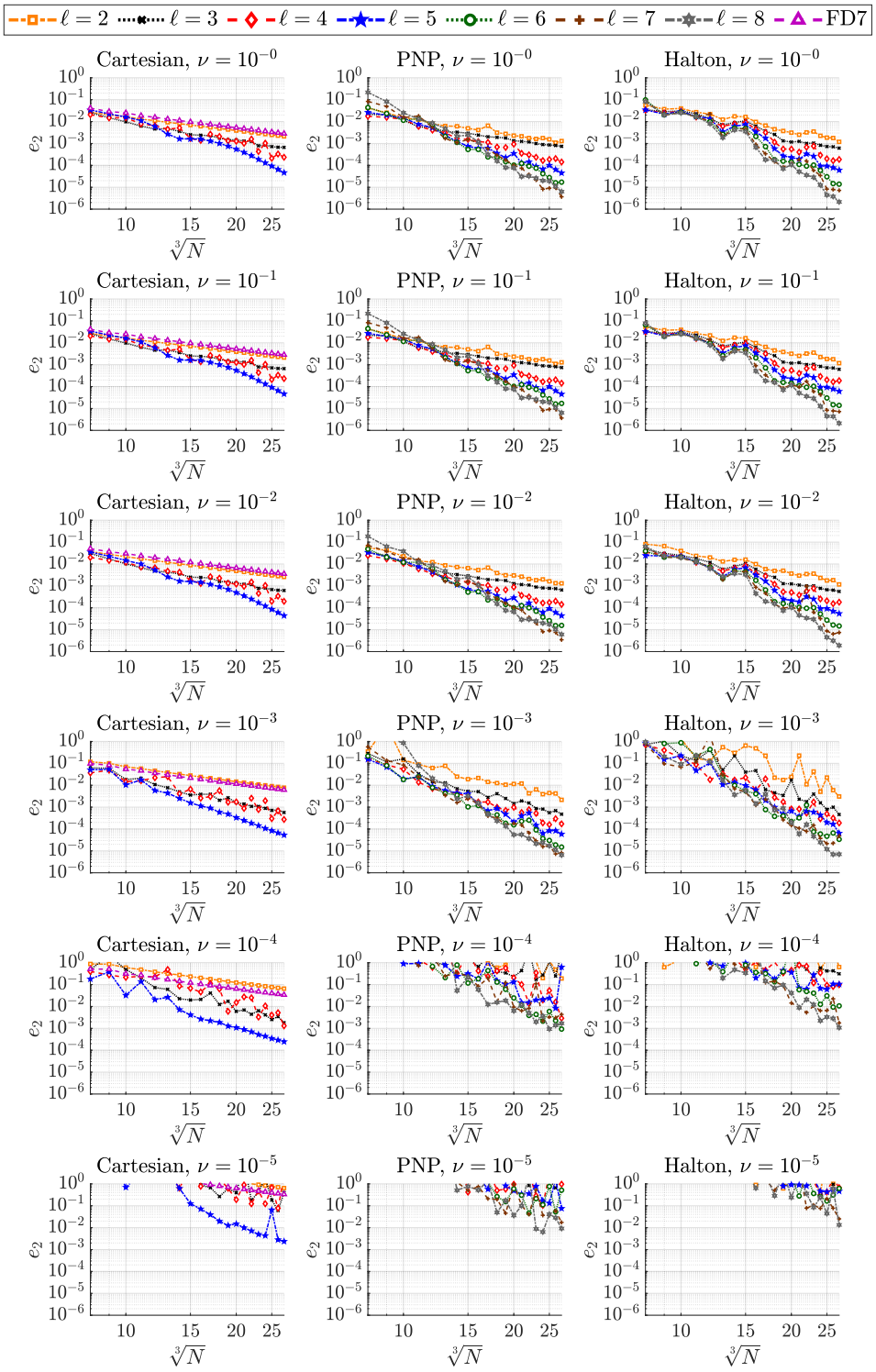

Figure 2.20: Errors (2.27) as a function of the number of nodes (Cartesian, PNP, Halton) for diffusion coefficients $\nu \in \{10^{-j} : j \in \{0, 1, 2, 3, 4, 5\}\}$ for PHS(k) generating function, degrees of polynomial augmentation $\ell \in \{3, 4, 5, 6, 7, 8\}$ (k and stencil size n are determined by (2.26) and (2.29), resp., for a fixed ℓ).

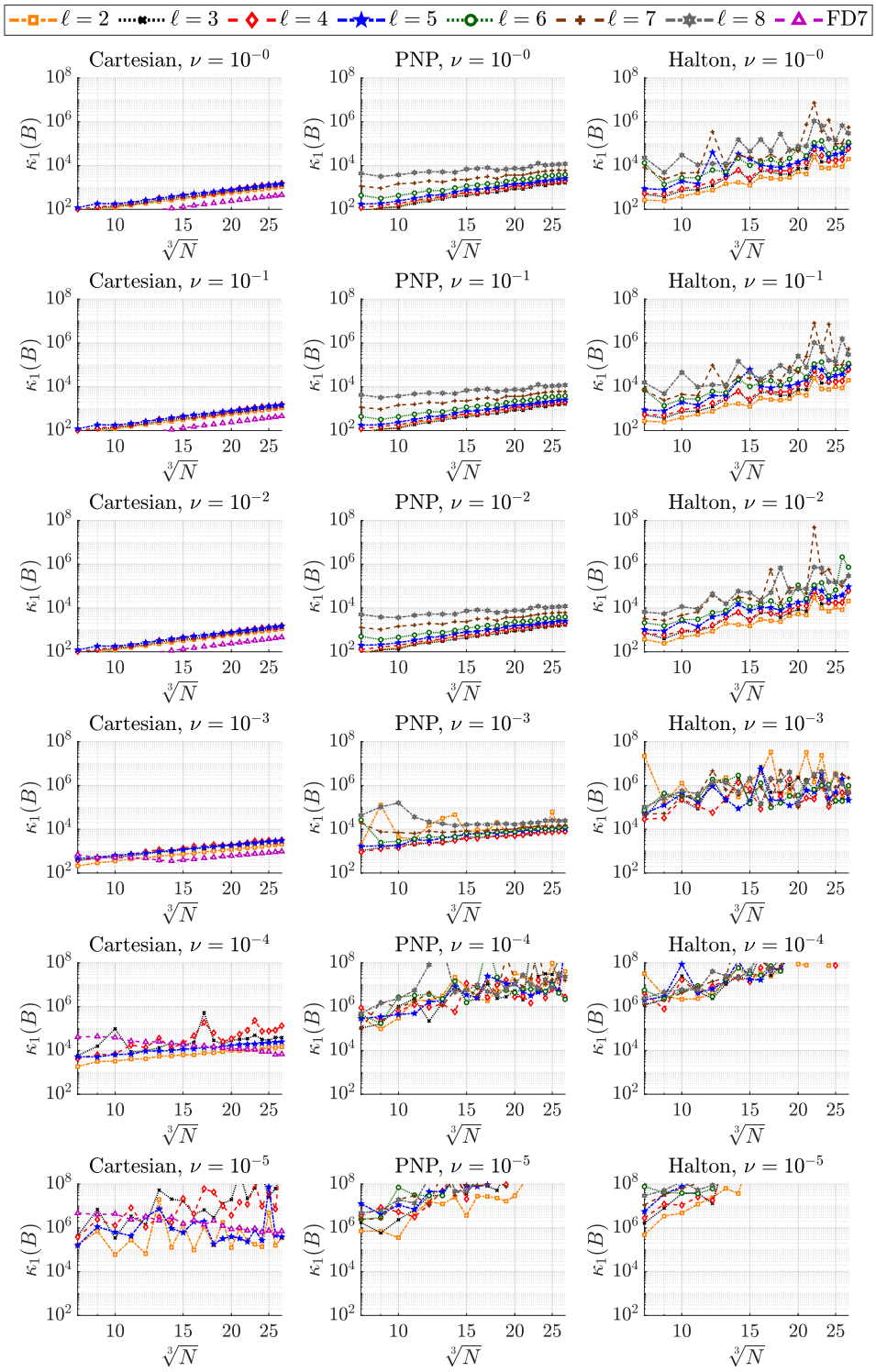

Figure 2.21: Condition numbers of the stiffness matrices as a function of the number of nodes (Cartesian, PNP, Halton) for diffusion coefficients $\nu \in \{10^{-j} : j \in \{0, 1, 2, 3, 4, 5\}\}$ for PHS(k) generating function, degrees of polynomial augmentation $\ell \in \{3, 4, 5, 6, 7, 8\}$ (k and stencil size n are determined by (2.26) and (2.29), resp., for a fixed ℓ).

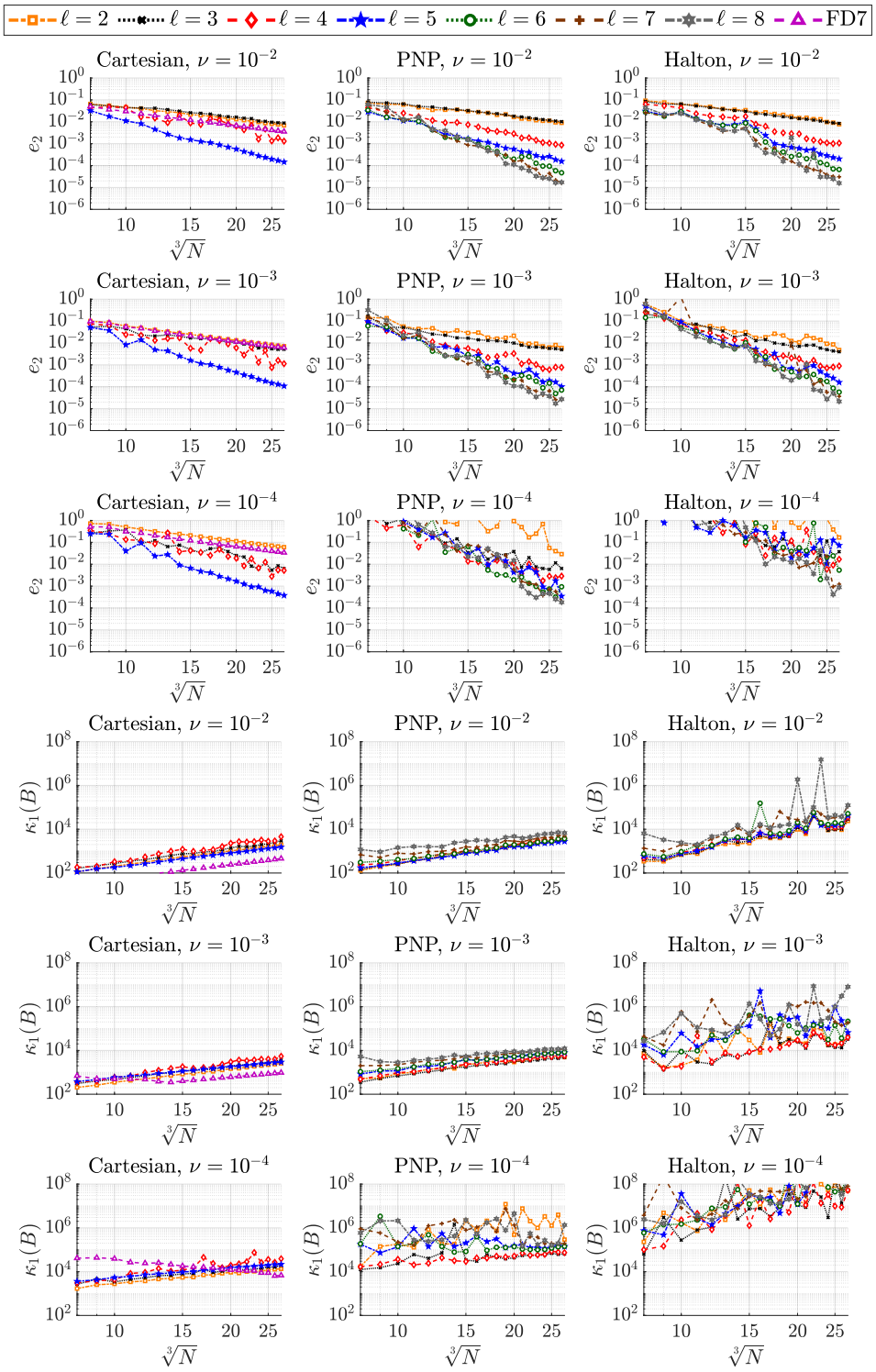

Figure 2.22: Errors (2.27) and condition numbers of the stiffness matrices as a function of the number of nodes (Cartesian, PNP, Halton) for diffusion coefficients $\nu \in \{10^{-j} : j \in \{2, 3, 4\}\}$ for PHS(k) generating function, degrees of polynomial augmentation $\ell \in \{3, 4, 5, 6, 7, 8\}$ (k and stencil size n are determined by (2.24) and (2.29), resp., for a fixed ℓ).

All of these figures consist of three columns of plots, showing

- results on Cartesian nodes (left column),
- results on PNP nodes (middle column),
- results on Halton nodes (right column).

Each row corresponds to a different (from top to bottom decreasing) diffusion coefficient ν.

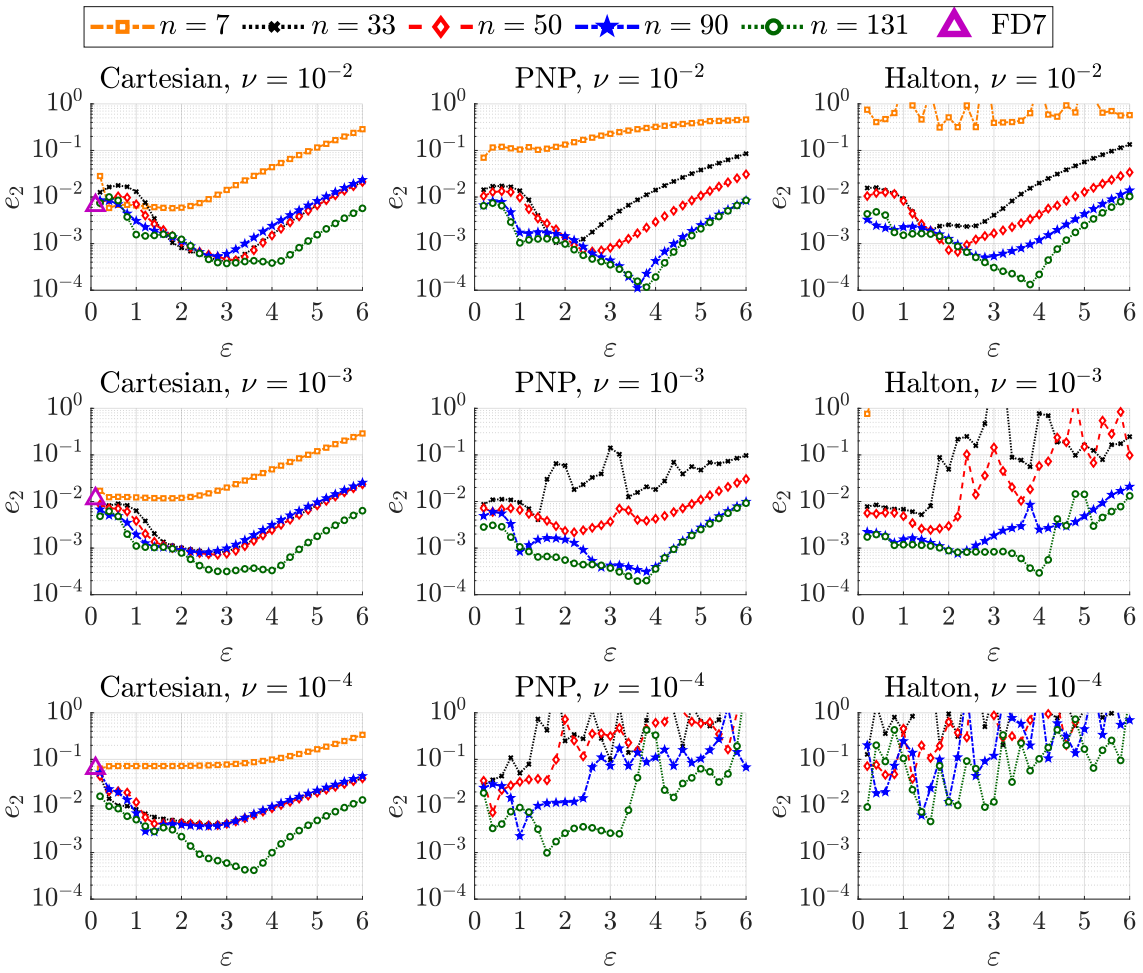

Figure 2.23: Errors (2.27) as a function of the shape parameter ε for diffusion coefficients $\nu \in \{10^{-j} : j \in \{2,3,4\}\}$, for the HYB($10^{-5}$) generating function, stencil sizes $n \in \{7, 35, 50, 90, 131\}$ and node distributions (Cartesian, PNP, Halton) with $N = 8000$ nodes and $N_I = 5832$ interior nodes.

Figure 2.20 demonstrates that the PHS setup can be used as well for the convection-diffusion problem as long as the diffusion coefficient ν is not too small. The errors (and oscillations in the errors) can increase with decreasing ν. These increases are especially large for $\nu \leq 10^{-4}$ and for $\nu = 10^{-3}$ if Halton nodes are combined with small degrees $\ell \leq 4$. Hence, upwinding (see Remark 2.11) would be needed for these setups. These

increases in the errors (and its oscillations) are the smallest for the Cartesian nodes and the largest for the Halton nodes. Nevertheless, the lowest errors for $\nu \geq 10^{-2}$ are obtained by Halton nodes with degree $\ell = 8$. On a Cartesian grid, the RBF-FD errors for a degree $\ell = 2$ are comparable to the 7-point-FD (also without upwinding) errors. The 7-point-FD errors are slightly larger than the errors for the degree $\ell = 2$ for $\nu \geq 10^{-2}$, and conversely for $\nu \leq 10^{-3}$.

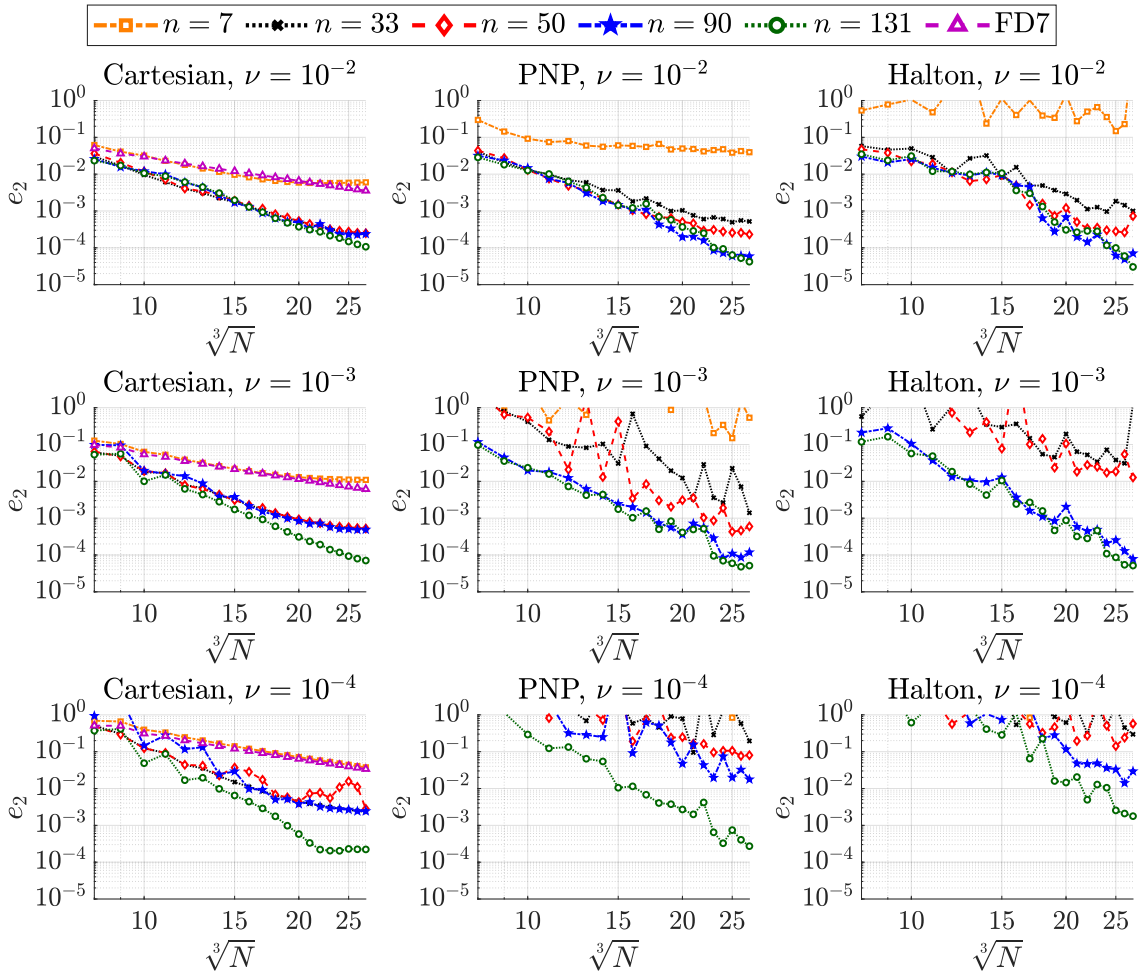

Figure 2.24: Errors (2.27) as a function of the number of nodes (Cartesian, PNP, Halton) for diffusion coefficients $\nu \in \{10^{-j} : j \in \{2, 3, 4\}\}$, for the HYB($10^{-5}$) generating function with polynomial augmentation of degree $\ell = 0$, $c = 0.15$ (2.28) and stencil sizes $n \in \{7, 33, 50, 90, 131\}$.

Figure 2.21 shows the condition numbers of the stiffness matrices for the setups tested in Figure 2.20. This underlines the observations that the PHS setup needs upwinding for $\nu \leq 10^{-4}$ since the condition numbers drastically increase with decreasing ν (such that they lie often above 10^8 for $\nu \leq 10^{-4}$). Cartesian nodes lead to the lowest condition numbers and Halton nodes lead to the highest condition numbers.

Figure 2.22 (as well as the further tests in this subsection) solely show results for diffusion coefficients $\nu \in \{10^{-j} : j \in \{2,3,4\}\}$ since smaller ν values lead to numerical instabilities whereas larger ν values lead to almost identical results as for the Poisson problem. This figure demonstrates that the PHS setup with (2.24) instead of (2.26) (i. e., with smaller PHS degrees) can, on one hand, significantly reduce the oscillations in the errors and in the condition numbers of the stiffness matrices. Additionally, these condition numbers can be significantly smaller for (2.24) than for (2.26). However, on the other hand the errors for $\nu \geq 10^{-2}$ are slightly larger for (2.24) than for the setup with (2.26). Furthermore, the difference between the results for polynomial degrees $\ell = 2$ and $\ell = 3$ is negligible for $\nu \geq 10^{-3}$ for (2.24) whereas $\ell = 3$ leads to lower errors than $\ell = 2$ in the setup with (2.26).

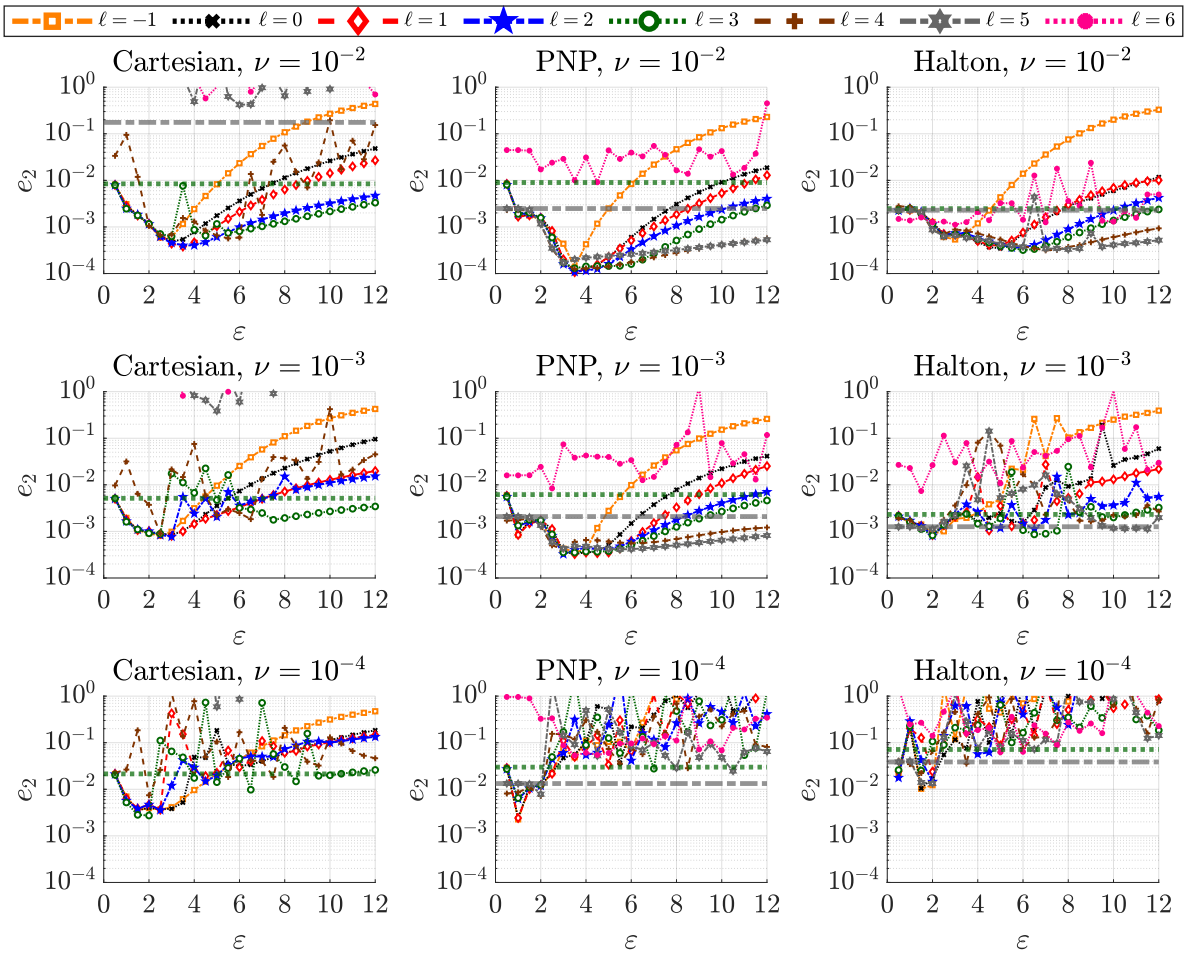

Figure 2.25: Errors (2.27) as a function of the shape parameter ε for diffusion coefficients $\nu \in \{10^{-j} : j \in \{2,3,4\}\}$, for the HYB($10^{-5}$) generating function, stencil size $n = 90$, polynomial augmentation of degree $\ell \in \{-1, 0, \ldots, 6\}$ and node distribution (Cartesian, PNP, Halton) with $N = 8000$ nodes and $N_I = 5832$ interior nodes. (Horizontal gray/green dashed lines show results for ε-independent PHS(2) with $\ell \in \{3, 5\}$.)

Figure 2.23 illustrates that (for the case without polynomial augmentation) the optimal shape parameter for HYB(10^{-5}) is almost the same as for the Poisson problem as long as the diffusion coefficient ν is not too small. The observations about increasing errors (and oscillations in the errors) with decreasing ν stated for PHS (see Figure 2.20) also hold for HYB. However, HYB seems to be slightly more unstable (than PHS with (2.24)) and upwinding would be needed for $\nu \leq 10^{-3}$ if PNP or Halton nodes are combined with smaller stencil sizes.

Figure 2.24 indicates that the observations seen in Figure 2.23 also hold, if the number of nodes N is increased and polynomial augmentation of degree $\ell = 0$ is used. Furthermore, it can be seen that larger stencil sizes seem to be more robust w. r. t. decreasing ν, e. g., Cartesian nodes achieve for $\nu = 10^{-2}$ almost identical approximation errors (2.27) for all stencil sizes $n \in \{50, 90, 131\}$, whereas the stencil size $n = 131$ leads for $\nu < 10^{-2}$ (and especially for larger numbers of nodes N) to significantly smaller errors than the stencil sizes $n \in \{50, 90\}$. The difference between the errors for $n \in \{33, 50\}$ and $n \in \{90, 131\}$ increases from $\nu = 10^{-2}$ to $\nu = 10^{-3}$ as well significantly for PNP and Halton nodes. The stencil sizes $n = 90$ and $n = 131$ lead for $\nu > 10^{-4}$ on PNP and Halton nodes to almost identical errors, whereas $n = 131$ leads for $\nu = 10^{-4}$ to significantly smaller errors than $n = 90$.

Figure 2.25 confirms that the observations seen in Figure 2.23 also hold, if larger degrees ℓ are augmented. Furthermore, it can be observed that Cartesian nodes with (larger degrees of) polynomial augmentation are especially sensitive to decreasing ν values, e. g., degrees $\ell \geq 3$, $\ell \geq 2$ and $\ell \geq 0$ lead to oscillations in the errors (2.27) for $\nu = 10^{-2}$, $\nu = 10^{-3}$ and $\nu = 10^{-4}$, resp.

2.4.7 Summary

In Subsection 2.4.2, we focused on the main difficulty of the (infinitely smooth) GA generating function: the selection of a shape parameter.

We concentrated in Subsection 2.4.3 on the PHS generating function. These tests illustrated problems for PHS with polynomial augmentation on Cartesian nodes for larger ℓ which required excessively large stencil sizes. For non-Cartesian grids, determining the stencil size via (2.29) and the PHS degree via (2.26) (or in case of numerical instabilities via the smaller (2.24)) seems reasonable in terms of accuracy and numerical stability. Furthermore, we illustrated that using shifting and scaling (Remark 2.14) to compute the stencil weights helps to show their "true" condition numbers such that numerical instabilities can be easier identified. However, using Remark 2.7 (i. e., solely shifting the stencil) seems to be sufficient in terms of numerical stability.

In Subsection 2.4.4, we showed tests for the HYB generating function which combines GA with PHS(2) and compared results with those obtained for several infinitely smooth RBFs and PHS. On one hand, we repeated the tests in Subsection 2.4.2 with HYB(γ) instead of GA. Our observations (in the setup with no or only small degrees of polynomial augmentation) agree with [85] that typically smaller errors and larger condition numbers

are obtained by decreasing γ. Additionally, we showed that HYB(γ), GA, IQ, IMQ and MQ generating functions can lead to comparable errors (for carefully selected shape parameters, see Tables 2.3 and 2.4 and (2.28) for general guidelines) and if γ is chosen according to (2.18).

We also focused on the role of polynomial augmentation. For GA, IQ, IMQ and MQ generating functions, typically a small degree ℓ (e. g., $\ell \leq 2$) is suggested. For HYB(γ), with a sufficiently large γ, we observed that increasing ℓ can lead to smaller errors than small ℓ (always paired with suitable shape parameters). Furthermore, we indicated that using Remark 2.7 (i. e., solely shifting the stencil) can be applied also to nonscalable RBFs and observed that this can reduce the errors and the condition numbers of the weight and stiffness matrices for larger degrees ℓ. Additionally, shifting the stencil to the origin can reduce the amount of floating point operations (as well as the associated numerical inaccuracies [111]) in the stencil weights computation (see Remark 2.8).

Therefore, we compared PHS and HYB generating functions with larger degrees of polynomial augmentation. These tests showed that $\ell > 3$ on Cartesian nodes is also not advisable for HYB. For $\ell \leq 5$ on PNP and Halton nodes, HYB can lead to smaller errors than PHS. For sufficiently large ℓ, HYB converges to PHS(2) as $\varepsilon \to 0$ which is reflected in the illustrated approximation errors (the condition numbers of the weight and stiffness matrices depend on γ). An advantage of using degrees $\ell \geq 3$ for HYB is the decrease of sensitivity w. r. t. the choice of γ or the shape parameter.

When it comes to the underlying nodes, one observes "PNP $\leq$ Halton $\leq$ random" where "$\leq$" relates the approximations errors (as well as condition numbers of the weight and stiffness matrices) obtained for those node sets. However, for PHS and HYB generating functions with larger degrees of polynomial augmentation, the differences are rather small, making HYB and PHS good options for adaptive node refinement and stencil selection as long as stencil sizes according to (2.29) (or larger) are used (for nearest neighbor stencils).

In Subsections 2.4.5 and 2.4.6, we illustrated that the qualitative results for Poisson's equation on a cube also hold for a wider class of problems. First the domain was changed from cube to sphere in Subsection 2.4.5, instead of that the differential operator was replaced in Subsection 2.4.6 to obtain the convection-diffusion equation.

Our recommendations for a good RBF-FD discretization setup using basic algorithms as described in this thesis may be summarized as follows:

- If Cartesian nodes are used, then a standard finite difference discretization or HYB with a small degree $\ell \in \{0, 1, 2\}$ of polynomial augmentation (and γ according to (2.18)) should be used.
- On unstructured nodes, PHS with polynomial augmentation offers a shape-parameter-independent approach. The required parameters ℓ, n, k may be determined via (2.20), (2.29) and (2.26) (or (2.24) if numerical instabilities occur), resp. The smallest errors are observed for large exponents and polynomial degrees $k = \ell = 8$ which require rather large stencils and benefit from shifting the stencil to

the origin. PHS setups are especially promising if a high accuracy approximation is wanted (i. e., if a large degree ℓ is used).

- On unstructured nodes, GA, MQ, IMQ, IQ or HYB with carefully finetuned parameters provide options to achieve similar accuracies as PHS with smaller stencils and hence may decrease the computational time or memory compared to the PHS setup. These generating functions are especially promising if a fast and low accuracy approximation is wanted since they do not rely on high degrees of polynomial augmentation.
- (Unstructured) PNP node sets often lead to the best results (smallest approximation error) among the various node sets included in our tests and are a promising option for more complicated domains Ω, adaptive node refinement or variable density node sets.

These recommendations almost certainly will have to be adjusted when more advanced algorithms for node generation [27, 113, 42, 91, 84, 30, 108, 101, 111] or stencil selection [23, 91, 24, 105, 92, 85, 22, 25, 99], stable versions of radial basis functions [44, 76, 71, 48, 129, 68] or computation of weights for deficient sets [21], etc., are used which are beyond the scope of this thesis.

2.5 Conclusion and outlook

RBF-FD provides a meshfree and relatively easy to implement approach for the discretization of partial differential equations. Its realization, however, involves a multitude of parameter choices, many of which (but by far not all) have been tested and illustrated in this chapter. Our conclusions for these have been summarized in Subsection 2.4.7. It turns out that several methods work well if parameters are chosen accordingly, and there is no clear winner (at least not among the tested variants). However, if parameters are not selected carefully, there are many "losers" leading to stagnation/divergence of the approximation error.

Our novel contributions of this chapter consist of

- a comprehensive view and comparison of basic RBF-FD versions from the literature;
- general recommendations for parameter setups in basic RBF-FD methods such as
  - a general recommendation to shift stencils to the origin if polynomials of large degrees are augmented;
  - a new scaling law for the stencil size and the degree of polynomial augmentation on Cartesian node distributions.

A straightforward extension of the work in this chapter would lie in the consideration of further RBF-FD variants, including stabilized versions, variable shape parameters, other than nearest neighbor stencils, further types of generating functions or their combina-

tions. It would furthermore be of interest to extend our study to other types of partial differential equations.

We have focused on the discretization process and resulting approximation errors of RBF-FD discretization and mostly left out considerations of computational complexity. The computation of stencil weights may be executed in parallel, hence an increase in stencil size (or polynomial augmentation degree) may not be a severe drawback w. r. t. computational time for the setup phase. However, it will lead to a less sparse and worse conditioned stiffness matrix and hence increase the computational time required for solving it (directly or iteratively). We focus in Chapter 3 on the design and analysis of iterative solvers and preconditioners for linear systems arising in RBF-FD discretizations.

3 Iterative solvers for RBF-FD

3.1 Introduction

The most time consuming part (i. e., more than 40% of the total computation time of the studied examples) in the RBF-FD approach (with PHS and polynomial augmentation) is the stencil weights computation for small problem sizes and the solution of the global linear system of equations for large problem sizes [61]. Hence, a significant speedup of the overall RBF-FD method can be achieved for smaller problem sizes by a speedup of the stencil weights computation, e. g., via overlapped RBF-FD method (a generalization of the RBF-FD method with nearest neighbor search, PHS generating function and polynomial augmentation [106, 109]). The idea of this approach is to change the computation of the stencil weights and the definition of the stiffness matrix such that the number of stencil weight computations is reduced in exchange for increased computational cost per stencil. This accelerates the setup of the stiffness matrix and is especially helpful for higher order methods (i. e., for larger degrees of polynomial augmentation). This overlapped RBF-FD method could be performed as well with the HYB instead of the PHS generating function. Another idea to accelerate the basic RBF-FD method with large degrees of polynomial augmentation (irrespective of the problem size) is the hybrid weighted least squares (WLS) - RBF-FD method [58] (i. e., a combination of the WLS approach and the RBF-FD approach) which is more stable than the WLS approach and cheaper than the RBF-FD approach. Furthermore, there are approaches which lead to a tradeoff between the setup time of the stiffness matrix and its solution such as more sophisticated algorithms for the node generation (see Subsection 2.3.1) or the stencil selection (see Subsection 2.3.2), e. g., the time spend due to a more sophisticated algorithm is well invested if the desired approximation accuracy can be achieved afterwards with fewer nodes or a smaller stencil size such that the overall computation time is reduced. A similar idea is refinement [111, 92, 122, 62, 59, 23, 91, 86] which can be subdivided into p-refinement, h-refinement and hp-refinement (and further types of refinement such as adaptive time-stepping [28]). The approximation order, i. e., the degree of polynomial augmentation (or the stencil size if they are not coupled by a formula such as (2.29)), is adaptively determined (and the discretization nodes are fixed) in a p-refinement approach such that a higher approximation order is only occasionally used if it is needed to achieve a desired overall approximation accuracy. An h-refinement approach works with exchanged roles, i. e., the discretization nodes are adaptively added or removed (and the approximation order, i. e., the degree of polynomial augmentation or the stencil size if they are not coupled by a formula such as (2.29) is fixed) such that a desired overall ap-

proximation accuracy is achieved. The *hp*-refinement approach combines *h*-refinement and *p*-refinement. However, a significant speedup of the overall RBF-FD method for larger problem sizes relies on a speedup of the solution of the global linear system of equations. One approach could be to use a mixed precision ansatz, since the condition numbers of the stencil matrices (and consequently the errors, see Subsection 2.2.3) in the stencil weight computations can be large (see Section 2.4). Hence, the loss in accuracy introduced by a lower floating point precision for the solution of the global linear system of equations than for the stencil weights could be relatively small. A different speedup approach lies in the development of more efficient solver setups (i. e., iterative solvers equipped with suitable preconditioners).

Currently there exists not much previous work about iterative solvers and preconditioners in the context of RBF-FD. Many applications of RBF-FD rely on standard iterative solvers, e. g., generalized minimal residual (GMRES) method [107, 82] (optionally with restarts [9]) and biconjugate gradient stabilized (BiCGstab) method [39, 88, 134, 135, 10, 61, 3, 32, 4], or on sparse direct solvers [61]. Iterative solvers in the context of RBF-FD are often combined with ILU preconditioners [39, 88, 107, 134, 135, 82, 10, 61, 3, 9, 32] (optionally with prior reordering, e. g., reverse Cuthill-McKee [10, 9, 39]). Moreover, a reverse Cuthill-McKee reordering and a customized preconditioner are used in [81]. Motivations to do reorderings include the reduction of fill-in, the improvement of the numerical stability of the decomposition [100], the optimization of the hardware (i. e., the memory) usage [13, 78] and the optimization of the convergence rate of the iterative solver [43]. However, iterative solvers with ILU preconditioners can be memory intensive and ineffective for large problem sizes N [81] since ILU preconditioners typically suffer from a number of iterations that increases with the problem size N [9], e. g., $\mathcal{O}(\sqrt[3]{N})$ [10]. Moreover, there exist preconditioners based on a (problem dependent) simplified version of the original PDE (e. g., with changed differential operator or boundary conditions) and combined with an ILU decomposition [9, 107] or based on the diagonal of an approximate Schur complement [109] (which is based on [12, Section 4]). Solvers relying on multilevel or multigrid ideas are introduced in [133, 94, 2, 69, 130].

In this chapter, we will study iterative solvers and preconditioners to speedup the solution of the global linear system of equations (hereinafter abbreviated as "the linear system"). We perform numerical tests with GMRES and BiCGstab solvers and preconditioners based on incomplete LU factorizations using sparse or hierarchical matrices [53] since they are one option to construct high accuracy preconditioners (with memory and computation time cost of almost linear complexity) such that numbers of iterations (almost) independent of problem size are achieved [73, 65]. Our focus for the numerical tests are the convection-diffusion equation and the PHS generating function. The novel contributions of this chapter consist of

- numerical tests and recommendations on the development of hierarchical matrix preconditioners to RBF-FD sparse matrices (see Section 3.4);
- the introduction of preconditioners that are based on an auxiliary RBF-FD stiffness matrix with a smaller stencil size (see Section 3.5).

The remainder of this chapter is organized as follows. We concentrate on general information about the numerical tests in Section 3.2. Section 3.3 is devoted to the solver choice and the performance of the ILU(0) preconditioner. In Section 3.4, we focus on hierarchical matrices and their application in the solution of sparse linear systems of equations arising in the RBF-FD method. Preconditioners based on an auxiliary RBF-FD stiffness matrix with a smaller stencil size are studied in Section 3.5. Our results are summarized in Section 3.6, along with recommendations for the setups of the iterative solver and the preconditioner, a conclusion and an outlook.

3.2 General information

In the following numerical tests, we concentrate on the PHS generating function with polynomial augmentation (i. e., for a fixed degree of polynomial augmentation $\ell \in \{2, 5, 8\}$, we use the generating function PHS(k) with k determined by (2.24) or (2.26) and the stencil size n given by (2.29), resp.) and use the following notation.

Notation 3.1. *We use a notation of the form PHS($k = \widetilde{k}$, $\ell = \widetilde{\ell}$) (with $\widetilde{k} \in \{2, 3, 4, 5, 8\}$ and $\widetilde{\ell} \in \{2, 5, 8\}$ according to the values in Table 3.1) to indicate that the generating function PHS($\widetilde{k}$) is used with polynomial augmentation of degree $\widetilde{\ell}$.*

We perform tests on Cartesian, PNP and Halton node distributions with up to $N = 90^3 = 729000$ total nodes and up to $N_I = 88^3 = 681472$ interior nodes. Results for random nodes are not presented since random nodes lead to higher errors than Halton nodes (see Subsection 2.4.4). Nevertheless, random nodes lead to qualitatively similar observations as Halton nodes with higher numbers of iterations and increased likelihood that the iterative solver does not converge (which can be explained by the higher condition numbers that were observed for random nodes in Subsection 2.4.4). Cartesian nodes are only used with degree $\ell = 2$ and the parameter values used are summarized in Table 3.1. The sparsity structures of the resulting stiffness matrices B (2.14) for PNP nodes with $N = 1728$ and $N_I = 1000$ are illustrated in Figure 3.1. Most of the stencil weights for PHS generating functions are relatively small, many of the few entries with large absolute values lie on the diagonal (i. e., correspond to a stencil center) which is also observed in [10].

ℓ	n	k (2.24)	k (2.26)
2	22	2	3
5	116	3	5
8	335	4	8

Table 3.1: Parameter values used in PHS setups.

Definition 3.2 (based on [100, 125]) introduces the basic ideas and notations of iterative solvers and preconditioners.

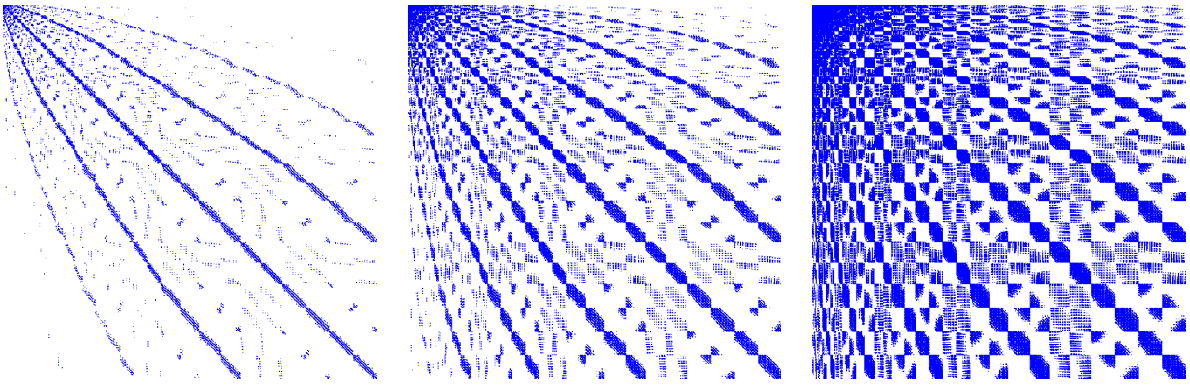

Figure 3.1: Sparsity structures of the stiffness matrices B (2.14) for PNP nodes with $N_I = 1000$ interior nodes (and $N = 1728$), polynomial degrees $\ell = 2, 5, 8$ (from left to right) and stencil sizes via (2.29), i. e., as in Table 3.1.

Definition 3.2. *Let $A \in \mathbb{R}^{n\times n}$, $b, x, y \in \mathbb{R}^n$, $\mathrm{TOL} > 0$, $m_{\max} \in \mathbb{N}$ and $x^{(m)} \in \mathbb{R}^n$ for all $m \in \mathbb{N}_0$. An iterative solver for the linear system of equations $Ax = b$ computes a sequence $(x^{(m)})_{m\in\mathbb{N}_0}$ until at least one of the following stopping criteria*

$$\|Ax^{(m)} - b\|_2 \leq \mathrm{TOL}\, \|Ax^{(0)} - b\|_2 \quad \text{or} \quad m = m_{\max}$$

is achieved. We call TOL *relative residual accuracy, $m_{\max}$ maximal number of iterations, $x^{(m)}$ the m'th iterate and $x^{(0)}$ start vector. A regular matrix $P \in \mathbb{R}^{n\times n}$ is called left* (3.1) *or right* (3.2) *preconditioner for $Ax = b$ if*

$$P^{-1}Ax = P^{-1}b \tag{3.1}$$

or

$$AP^{-1}y = b, \quad x = P^{-1}y \tag{3.2}$$

is solved instead.

The test problem is the convection-diffusion equation (see Subsection 2.4.6). The corresponding errors (2.27) and condition numbers of the stiffness matrices are shown in Figures 2.20 - 2.22. These figures illustrate that the RBF-FD approach (without upwinding) becomes unstable if the diffusion coefficient ν becomes too small. We observed (tests not included here) that the differences in terms of the performance of the iterative solvers and preconditioners between the Poisson problem and the convection-diffusion equation diminish for increasing diffusion coefficient ν. However, a decrease in ν increases the likelihood that basic solvers and preconditioners (e. g., BiCGstab without preconditioning or with ILU(0), Jacobi, Gauss-Seidel, backward or symmetric Gauss-Seidel preconditioner) will fail to converge (or will lead to slow convergence). Therefore, we concentrate on the case with $\nu = 10^{-3}$. On one side, the RBF-FD method (without upwinding) is still stable (if an appropriate node distribution is used, e. g., Cartesian,

PNP or Halton nodes, whereas random nodes can lead to a breakdown in the stencil weights computation). On the other side, basic solvers (especially BiCGstab) and preconditioners can fail to converge (or can lead to slow convergence).

We perform tests with left preconditioned BiCGstab (LPBiCGstab), left preconditioned GMRES (LPGMRES) and right preconditioned GMRES (RPGMRES) solvers and will only write BiCGstab and GMRES if we do not emphasize how or if a preconditioner is applied. The computed solution u is given in this chapter by the final iterate of the iterative solver, i. e., this iterate fulfills a stopping criterion (see Definition 3.2). Hence, the approximation errors (2.27) originate from divergence of the iterative solver (i. e., the solver performed $m_{\max}$ iterations) or from highly ill-conditioned preconditioners (or from highly ill-conditioned stiffness matrices which are here usually not a problem, see Figures 2.20 - 2.22). As a stopping criterion, we use the relative residual accuracy $\mathrm{TOL} = 10^{-13}$. This relatively small residual accuracy is of the order $10^{-15} \cdot 90^2 \approx 8 \cdot 10^{-12}$ which is the smallest achievable error in double precision w. r. t. floating point arithmetic for a discretization in $d = 3$ dimensions with 90^3 nodes and (approximate) internodal distance of $h \approx N^{-1/d}$ (2.17), hence the difference in the approximation error (2.27) between different linear system solvers is usually negligible. In the literature, the residual accuracy TOL often satisfies $\mathrm{TOL} \in [10^{-11}, 10^{-6}]$ [10, 9, 39, 82, 88, 134, 135] whereas an even smaller residual accuracy of $\mathrm{TOL} = 10^{-15}$ is used in [60, 61]. If the approximation error (2.27) is known (or estimated) a priori, then this knowledge could be used to adaptively select the relative residual accuracy.

The start vector is the zero vector and the maximal number of iterations is $m_{\max} = 10^4$. The actual number of iterations is typically significantly smaller, but we want to investigate in which cases a worse preconditioner (in combination with a higher number of iterations) can lead to an overall smaller solution time than a better preconditioner with a higher setup time. An ILU(0) preconditioner without reordering and different $\mathcal{H}$-LU preconditioners are tested. Results describing the setup time (in seconds) and the needed memory (in KB) of the preconditioner, the number of iterations, the actual solution time (in seconds) of the linear system, accumulated computation times, and errors (2.27) (i. e., we do not distinguish approximation errors from iteration errors in the plots) are shown. Many plots are scaled by $1/(N \log(N))$ or $1/(N \log^2(N))$ to illustrate the asymptotic complexities for increasing problem sizes N (see Remark 3.3). However, this hides the significant differences in the memory and computation cost of larger problem sizes since what may appear as a small difference in the plot may correspond to a large difference (see Table 3.2). Furthermore, the plots show not every marker to avoid too cluttered plots.

Remark 3.3. *An asymptotic complexity $\mathcal{O}(C(N))$ of a computation time w. r. t. the problem size N can be heuristically determined by plotting the computation time scaled by $1/C(N)$. If the plotted values appear to be bounded by a constant for $N \to \infty$, this supports the complexity assumption $\mathcal{O}(C(N))$. This thesis includes only plots with factors $C(N) = N \log(N)$ and $C(N) = N \log^2(N)$ but also statements with other complexities that we deduced from plots not shown here.*

N	$1/(N\log(N))$	$1/(N\log^2(N))$
$10^3 = 1000$	$\approx 3 \cdot 10^{-4}$	$\approx 1 \cdot 10^{-4}$
$70^3 = 343000$	$\approx 5 \cdot 10^{-7}$	$\approx 1 \cdot 10^{-7}$
$90^3 = 729000$	$\approx 2 \cdot 10^{-7}$	$\approx 4 \cdot 10^{-8}$

Table 3.2: Some scaling factors to illustrate the asymptotic complexities.

All computation time measurements were done for computations in double precision on a single core of a computer with an *Intel(R) Xeon(R) CPU E5-2665 0 @ 2.40 GHz* processor and 16 GB of DDR3 memory. Each time measurement represents one computation (i. e., no average computation times are determined) since the differences between different runs of the same computation are typically relatively small compared to the differences between the compared setups (especially if we focus on the asymptotic complexities). The default options of the H2Lib library [52] were used, except that we turned on OPT (i. e., used the compiler flags *-O3 -march=native -funroll-loops*), added the compiler flag *-DNDEBUG* and turned off DEBUG and USE_CAIRO (i. e., compiled without *-g* and build without plot functions) with *gcc (SUSE Linux)* 7.5.0. For consistency, we added the compiler flags *-march=native -funroll-loops -DNDEBUG* to the default options of the ANN library [89] and used *g++ (SUSE Linux)* 7.5.0. Our own code was included in the H2Lib library when it did not rely on the ANN library, and otherwise compiled with *gcc (SUSE Linux)* 7.5.0 with compiler flags *-O3 -march=native -funroll-loops -DNDEBUG*.

3.3 Solver choice and ILU(0) preconditioners

The focus of this section is comparing different solvers (LPBiCGstab, LPGMRES and RPGMRES) in combination with ILU(0) preconditioners. Only ILU(0) preconditioners without reordering are utilized in this thesis (as announced in Section 3.2), i. e., the ordering depends on the implementation, namely the order in which the nodes are generated. Nevertheless, we tested (not included here) different kinds of reorderings which were mainly beneficial for ILU preconditioners with fill-in such as ILU(p) preconditioners with $p > 0$ [100] (which are not included here). The additional computation cost of a reordering was usually not amortized for ILU(0) preconditioners since they do not suffer from fill-in. The results of this section are presented through the following figures.

- Figure 3.2: Influence of the problem size N on the computation time, the numbers of iterations and the errors (2.27) for the PHS(k) generating function where k is determined by (2.26).
- Figure 3.3: Influence of the problem size N on the computation time, the numbers of iterations and the errors (2.27) for the PHS(k) generating function where k is determined by (2.24).

All of these figures consist of three columns of plots, showing

- scaled computation times (left column),
- numbers of iterations (middle column),
- errors (2.27) (right column).

Each row corresponds to a different node distribution (Cartesian, PNP, Halton) or a different degree $\ell \in \{2, 5, 8\}$.

Figure 3.2 focuses on the PHS(k) generating function where k is determined by (2.26). The first three rows show results on different node distributions (Cartesian, PNP and Halton) with degree $\ell = 2$. The last two rows include results on PNP nodes with larger degrees $\ell \in \{5, 8\}$. Each color/marker represents a different solver (LPBiCGstab, LPGMRES and RPGMRES). The tests for degree $\ell = 2$ for Halton nodes and for degree $\ell = 8$ are only performed up to $N = 61^3 = 226981$ nodes since the tests with GMRES are extremely time consuming if the maximal number of iterations 10^4 is reached. Additionally, these setups do not lead to reliable approximations (i. e., the errors (2.27) are usually above 10^{-1}).

The left column shows the computation times (in seconds) to solve the linear system scaled by $1/(N \log(N))$ (i. e., the inverse of the complexity of the setup of the linear system [61]). This is the only column that uses the additional colors/markers "RBF-FD" and "ILU0" to illustrate the setup times of the linear system (without the setup time of the node distribution) and of the ILU(0) preconditioner, resp. It can be observed that the setup time of the ILU(0) preconditioner is significantly lower than the setup time of the linear system for all degrees and for all node distributions. Their asymptotic complexities are similar, whereas the constants in these complexities increase with the degree ℓ (i. e., with the stencil size n (2.29), see Table 3.1 and Figure 3.1, since an increase in the stencil size usually increases the amount of nonzero entries in the stiffness matrix). The solution times can be lower than the setup times of the linear system and of the preconditioner. However, the solution times typically increase with the problem size since the condition numbers of the stiffness matrices, which are illustrated in the fourth row of Figure 2.21, typically increase as well with the problem size, whereas the solution times are oscillatory for PNP nodes except for large N for the degrees $\ell \in \{2, 5\}$ (i. e., in the cases without oscillations the complexity of the solution of the linear system is significantly higher than the complexities of the setup of the linear system and of the preconditioner). Furthermore, it can be observed that the solution time for setups with (almost) identical numbers of iterations (e. g., if BiCGstab does not converge) increases with the degree ℓ since an increase in the degree ℓ usually increases the amount of nonzero entries in the stiffness matrix, see Figure 3.1. The impact of the choice of the solver (if BiCGstab converges) on the overall computation time is negligible for smaller problem sizes. Nevertheless, BiCGstab (if it converges) should be typically preferred over GMRES since it is significantly faster than GMRES especially for larger problem sizes. The LPGMRES results are often invisible since they are overlaid by the RPGMRES results (except for the case with $\ell = 8$). This demonstrates that left and right preconditioning often lead to similar results. The node distribution influences the condition number of the stiffness matrix which significantly influences the solution

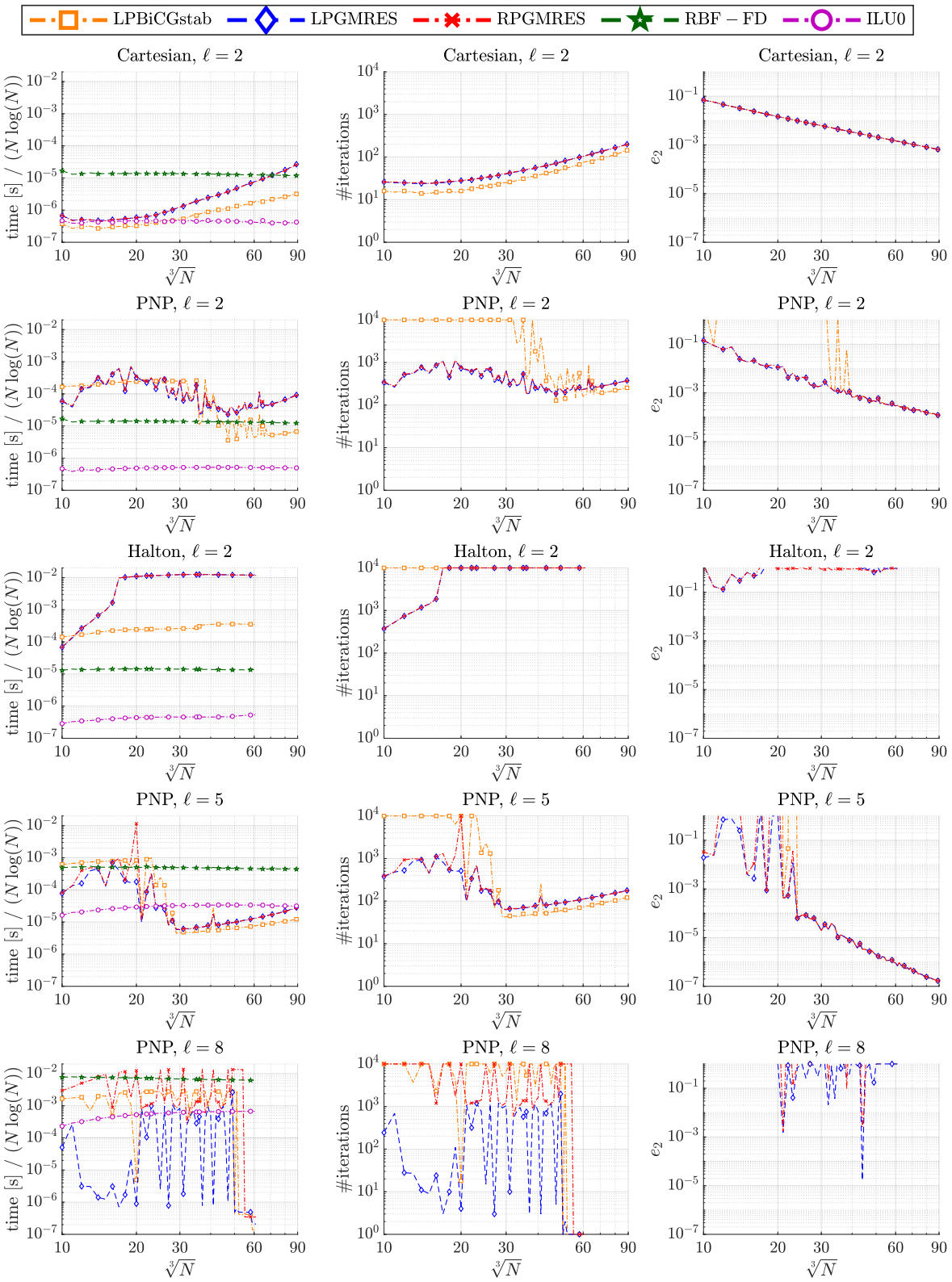

Figure 3.2: Scaled computation times (in seconds), numbers of iterations and approximation errors (2.27) as a function of the number of nodes for the PHS(k) generating function where k is determined by (2.26) for polynomial degrees $\ell \in \{2, 5, 8\}$ and several node distributions (Cartesian, PNP, Halton).

time (i. e., the lowest and the highest solution times correspond to Cartesian and Halton nodes, resp.), whereas its influence on the setup times of the linear system and of the preconditioner is negligible.

The middle column shows the numbers of iterations of the iterative solvers. This illustrates that the oscillations in the solution times for PNP nodes are based on oscillations in the numbers of iterations. The cases without oscillations confirm that the numbers of iterations increase with the problem size [9, 10]. GMRES always converges for Cartesian nodes and PNP nodes with degree $\ell = 2$ and BiCGstab fails to converge always for Halton nodes and often for PNP nodes. The difference between LPGMRES and RPGMRES is, once again, invisible (except for the cases with degrees $\ell \in \{5, 8\}$). GMRES and BiCGstab typically lead to similar numbers of iterations (slightly smaller for BiCGstab) if the solver converges and the degree of polynomial augmentation is not too large (i. e., for $\ell < 8$). Nevertheless (in contrast to Figure 3.2 with the largr PHS degrees (2.26)), the difference between LPGMRES and RPGMRES is invisible for the degree $\ell = 5$ and often invisible for the degree $\ell = 8$. Cartesian nodes need the fewest iterations and always lead to convergence of the solver.

The right column illustrates the approximation errors (2.27) to indicate whether the computed approximate solutions are reliable. This demonstrates that the relative residual tolerance 10^{-13} is small enough and the maximal number of iterations 10^4 is large enough for the degrees $\ell < 8$ to see no difference in the plots of the approximation errors between the computed solutions of the different solvers (i. e., the iteration errors are smaller than the approximation errors) if the solvers converged without oscillations in the numbers of iterations. However, the differences in the approximation errors between the computed solutions can be large for PNP nodes (even if the solvers converged, e. g., larger by more than a factor of 10). Furthermore, the solutions for the larger degree $\ell = 8$ can correspond to large approximation errors, even if the corresponding number of iterations is small.

Figure 3.3 is similar to Figure 3.2 except that k is determined by (2.24) (i. e., the sparsity structures of the stiffness matrices are unchanged since they are independent of k, see Figure 3.1) and the tests for Halton nodes are performed up to $N = 90^3$ nodes. This figure demonstrates that the smaller PHS degrees given by (2.24) hardly change the results for Cartesian nodes whereas they significantly improve the numerical stability for PNP and Halton nodes. This observation can be explained by the underlying condition numbers. Namely, the condition numbers of the stencil matrices are independent of the differential operator $\mathcal{L}$ and Figure 2.8 illustrates that higher PHS degrees often lead to higher condition numbers of the stencil matrices. The condition numbers of the stiffness matrices (corresponding to Figures 3.2 and 3.3) are shown in Figures 2.21 and 2.22, resp. This demonstrates that larger PHS degrees typically lead to higher condition numbers of the stiffness matrices (and this increase is especially large for the polynomial degree $\ell = 8$). Furthermore, Halton nodes typically lead to the largest condition numbers of the stiffness matrices.

The solution times and the numbers of iterations can be still oscillatory for Halton

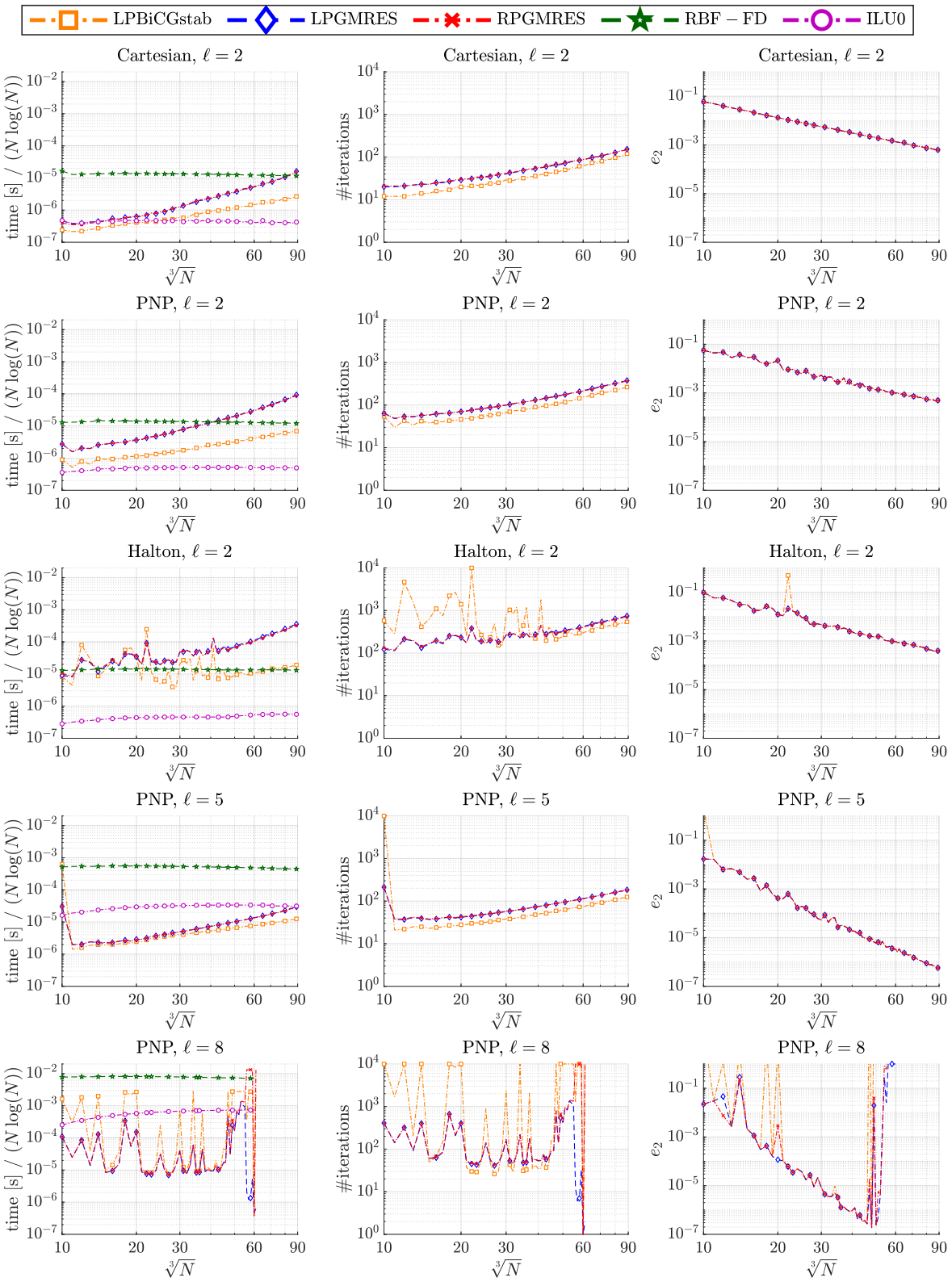

Figure 3.3: Scaled computation times (in seconds), numbers of iterations and approximation errors (2.27) as a function of the number of nodes for the PHS(k) generating function where k is determined by (2.24) for polynomial degrees $\ell \in \{2, 5, 8\}$ and several node distributions (Cartesian, PNP, Halton).

nodes and are oscillatory for the degree $\ell = 8$. However, GMRES always converges for $\ell < 8$ and BiCGstab fails to converge once for Halton nodes with $\ell = 2$ and once for PNP nodes with $\ell = 5$, whereas all solvers fail to converge for several problem sizes for degree $\ell = 8$. Cartesian nodes need the fewest iterations, whereas Halton nodes need the most iterations (i. e., a decrease of the regularity of the node distribution increases the numbers of iterations). Nevertheless, it can be observed that the relative residual tolerance 10^{-13} is small enough and the maximal number of iterations 10^4 is large enough for the degrees $\ell < 8$ to see no difference in the plots of the approximation errors between the computed solutions of the different solvers (i. e., the iteration errors are smaller than the approximation errors) if the solvers converged. However, the differences in the approximation errors between the computed solutions can be still large for the degree $\ell = 8$ (even if the solvers converged, e. g., larger by more than a factor of 10 between LPGMRES and RPGMRES for $N = 20^3$, although Figure 2.22 shows that the condition number of the corresponding stiffness matrix is only approximately 10^4 (and the LU solver computes an error (2.27) of approximately 10^{-4}, which is similar to the error of the LPGMRES solver)). Large errors are occurring for all solvers for approximately $N \geq 48^3$. Furthermore, the solutions for the larger problem sizes (e. g., $N \approx 60^3$) correspond to large approximation errors, although the numbers of iterations are small. Similar observations where RBF-FD discretizations become unstable if a large degree is combined with a large N are reported in [60].

The results in this section confirm the results in [107] that larger PHS degrees than given by (2.24) can significantly increase numerical instabilities (especially if PNP or Halton nodes are used). Furthermore, the results in [61] are confirmed, i. e., the most time consuming part in the RBF-FD approach is either (for small problem sizes) the setup of the linear system or (for large problem sizes) the solution of the linear system. The breakeven point in terms of N depends on the degree ℓ (or here, equivalently, the stencil size), the PHS degree, the type of the node distribution, and the solver. Further observations are that, on one hand, ILU(0) preconditioners can lead to fast and reliable results, especially if more regular node distributions (e. g., Cartesian or PNP nodes) and smaller degrees (e. g., polynomial degree $\ell \leq 5$ and PHS degree given by (2.24)) are used. On the other hand, the numbers of iterations usually increase with the problem size N and the irregularity of the node distribution. Additionally, if the degree ℓ of polynomial augmentation or the PHS degree (2.26) is large, the results may not be reliable at all. Further tests (not included here) indicate that the likelihood of unreliable results increases with ℓ, k and with a decrease in the regularity of the node distribution. The degree ℓ up to which the computed solutions are reliable depends on the node distribution and the PHS degree. This problem occurs since in floating point arithmetic the L and U factors of the ILU(0) decomposition can be almost singular (i. e., with condition numbers of approximately 10^{16} in double precision) even though the condition number of the decomposed stiffness matrix is smaller than 10^5 (see Figures 2.21 and 2.22 and Subsection 2.2.3). However, the lower bound for the error (2.27) w. r. t. the condition number of the decomposed stiffness matrix is 10^{-11} (see Subsection 2.2.3). Hence, a theoretical achievable discretization error is not achieved because of a

poor preconditioner. Furthermore, this can be tricky in practical applications since the convergence of the solver could be interpreted as the reliability of the computed solution. Therefore, in the following, we focus on the construction of more reliable preconditioners (ideally with problem size independent numbers of iterations) and concentrate mainly on PHS degrees given by (2.24).

3.4 Hierarchical matrices

This section deals with hierarchical matrices. Subsection 3.4.1 introduces hierarchical matrices. In Subsection 3.4.2, we summarize results for $\mathcal{H}$-matrix preconditioners for other sparse matrices (e. g., from the FE or FD methods). We explain in Subsection 3.4.3 when and why $\mathcal{H}$-matrices are useful in the RBF-FD setting and how these $\mathcal{H}$-matrices should be constructed. In Subsection 3.4.4, we discuss how to setup an $\mathcal{H}$-matrix preconditioner in the library H2Lib [52] for the sparse stiffness matrix in the RBF-FD approach. We perform numerical tests with polynomial degree $\ell = 2$ on different node distributions (i. e., Cartesian, PNP, Halton) in the Subsection 3.4.5 to illustrate the performance of $\mathcal{H}$-LU preconditioners. In Subsection 3.4.6, we focus on PNP nodes and study whether the results from the previous Subsection 3.4.5 generalize to larger polynomial degrees $\ell \in \{5, 8\}$.

3.4.1 Introduction to hierarchical matrices

We introduce $\mathcal{H}$-matrices (based on [53]) and focus on the aspects and theoretical results which are important in understanding the following results. The basic idea in the setup of an $\mathcal{H}$-matrix is to transform a matrix to another format such that knowledge about the underlying problem and the "structure" of the matrix can be exploited to reduce the cost of memory and arithmetic operations. $\mathcal{H}$-matrices were introduced for the approximation of full matrices (see Definition 3.4) but can be also useful in the context of sparse matrices (see Subsection 3.4.2). If $M \in \mathbb{R}^{n\times n}$, $n \in \mathbb{N}$, is a general square full matrix, then the cost of memory and arithmetic operations is at least $\mathcal{O}(n^2)$. The aim for the construction of an $\mathcal{H}$-matrix is to approximate M by (or transform M into) an $\mathcal{H}$-matrix such that the cost of memory and arithmetic operations is reduced to almost linear complexity (i. e., $\mathcal{O}(n \log^\alpha(n))$ with $\alpha \geq 0$ and independent of n, see Remark 3.18).

Definition 3.4. *Let I, J be finite index sets with $I \neq \emptyset \neq J$. We define $\mathbb{R}^I$ as the set of all mappings $I \to \mathbb{R}, i \mapsto x_i$ and $\mathbb{R}^{I\times J}$ as the set of all linear mappings $\mathbb{R}^J \to \mathbb{R}^I$ and write $x = (x_i)_{i\in I} \in \mathbb{R}^I$ and $M = (M_{i,j})_{i\in I, j\in J} \in \mathbb{R}^{I\times J}$. The notation $\mathbb{R}^n$ is a general term for all $\mathbb{R}^I$ with $\#I = n$ (# denotes the cardinality, i. e., the number of elements in the set) and $\mathbb{R}^{n\times m}$, $\mathbb{R}^{I\times m}$ as well as $\mathbb{R}^{n\times J}$ are defined analogously. A matrix $M \in \mathbb{R}^{I\times J}$ is called full (or dense) matrix if all $\#I \cdot \#J$ entries have to be saved to represent M. We write $M \in \mathcal{F}$ (or $M \in \mathcal{F}(I \times J)$) if M is a full matrix.*

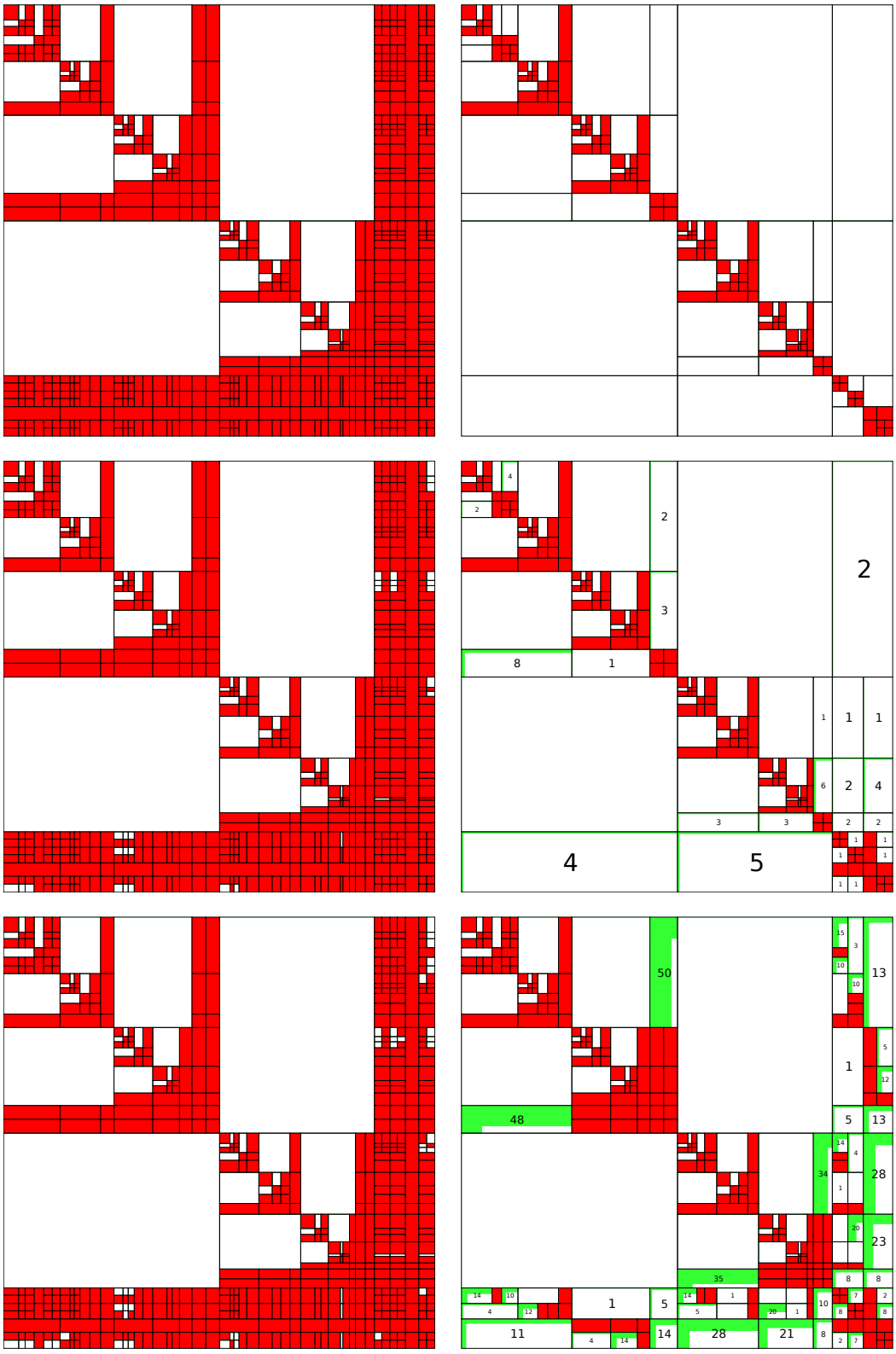

Figure 3.4: Examples of $\mathcal{H}$-matrices for the stiffness matrix shown in the left column of Figure 3.1 (see Remark 3.19 for setup details of these $\mathcal{H}$-matrices). Exact representations in the left column and approximations in the right column.

Figure 3.4 (created by the H2Lib function draw_cairo_hmatrix [52]) shows examples of $\mathcal{H}$-matrices (see Remark 3.19 for setup details). The left column of Figure 3.4 consists of exact representations of the stiffness matrix in the left column of Figure 3.1, whereas the right column exhibits approximations of the same matrix. This figure illustrates the basic idea in the construction of an $\mathcal{H}$-matrix, i. e., to hierarchically partition the row and column index sets I and J, resp., (here $I = J = \{1, \dots, n\}$ with $n = N_I$) such that some matrix blocks (see Definition 3.5) are saved as low-rank approximations (white and green blocks, see Definition 3.6), whereas the other matrix blocks are saved as full matrices (red blocks). The numbers indicate the ranks (which should be ideally as low as possible) of the low-rank approximations (with rank $r = 0$ in white blocks). The memory cost of each of the low-rank factors (relative to the memory cost of the respective full matrix block) is visualized with green.

Definition 3.5. *Let I, J be finite index sets with $I \neq \emptyset \neq J$, $p \in \mathbb{N}$ and $M \in \mathbb{R}^{I \times J}$. The set $P = \{b_1, \dots, b_p\} \subseteq \mathcal{P}(I \times J) \setminus \{\emptyset\}$ (with $\mathcal{P}$ denoting the power set) is called a block partition of $I \times J$ if*

1. *for all $i \in \{1, \dots, p\}$ there are sets $I_i \subseteq I, J_i \subseteq J$ such that $b_i = I_i \times J_i$ and*
2. *$\dot{\bigcup}_{j \in \{1,\dots,p\}} b_j = I \times J$.*

Each $b \in P$ is called an (index) block, the submatrix $M|_b := (M_{i,j})_{(i,j) \in b} \in \mathbb{R}^b$ corresponding to the block b is called a matrix block and the matrix M is called block matrix. We write $M|_b \in \mathcal{F}(b)$ if $M|_b$ is a full matrix.

Definition 3.6. *Let I, J be finite index sets with $I \neq \emptyset \neq J$, $r \in \mathbb{N}_0$ and $M \in \mathbb{R}^{I \times J}$. We call M a rank-r matrix and write $M \in \mathcal{R}(r)$ (or $M \in \mathcal{R}(r, I \times J)$) if $\operatorname{rank}(M) \leq r$ holds. Then, for $r > 0$, there exist $A \in \mathbb{R}^{I \times r}$ and $B \in \mathbb{R}^{J \times r}$ such that $M = AB^T$. If $\|M - AB^T\|_2 \leq \varepsilon \|M\|_2$ holds for some $\varepsilon \geq 0, A \in \mathbb{R}^{I \times r}$, $B \in \mathbb{R}^{J \times r}$, we call $R := AB^T$ a low-rank approximation of M (of rank r with relative tolerance ε).*

The definition of an $\mathcal{H}$-matrix needs the following ingredients. The hierarchical block structure of an $\mathcal{H}$-matrix is represented by a block cluster tree $T(I \times J)$ (see Definition 3.8). Such a block cluster tree $T(I \times J)$ is a cluster tree (or set decomposition tree, see Definition 3.8) corresponding to the index set $I \times J$. Hence, a block cluster tree $T(I \times J)$ is basically a labeled tree (see Definitions 3.7) with labels that are subsets of $I \times J$. Important reasons for the computation of a block cluster tree $T(I \times J)$ are: the leaves $\mathcal{L}(T(I \times J))$ (see Definitions 3.7) yield a (block) partition of $I \times J$ (which is needed to define an $\mathcal{H}$-matrix, see Definition 3.9) and the computation cost has an almost linear complexity. The sparsity constants (see Definition 3.9) can be used to bound the memory cost of an $\mathcal{H}$-matrix (and with that the cost of $\mathcal{H}$-arithmetic involving this $\mathcal{H}$-matrix, see [53, Section 6.3] for more details). A size function $size_{T(I \times J)}$ and an admissibility condition *adm* (see Definition 3.9) are utilized to decide the stopping point of the partition of blocks and which blocks should be saved as full matrices and which should be saved as rank-r matrices. These functions determine if a partition P is admissible, and if P is not admissible, this partition will be refined until an admissible

partition arises. The sets near-field P^- and far-field P^+ (see Definition 3.9) describe, for an admissible partition P, which blocks correspond to full or rank-r matrices, resp. The partition P can be coarsened afterwards (see Definition 3.15 and right column of Figure 3.4). An admissibility condition *adm* depends on a priori information about the underlying problem (see Definition 3.10 for examples, each row in Figure 3.4 has been produced using a different admissibility condition). For sparse matrices this function *adm* is often the weak admissibility condition (see Definition 3.10), and if a nested dissection ordering is used, then the knowledge that certain zero blocks remain zero in the LU decomposition (see Figure 3.4, Definition 3.16 and Remark 3.17) should be included [50, 72].

Definition 3.7. *Let $V \neq \emptyset$ be a finite set with $v, w \in V$, $\mathcal{P}(V)$ the power set, $S\colon V \to \mathcal{P}(V)$ a mapping and $k \in \mathbb{N}_0$. Then, we call*

1. *S son mapping;*
2. *v father of w and w son of v if $w \in S(v)$;*
3. *$(v_0, v_1, \ldots, v_k)$ path (of length k from v_0 to v_k), v_k successor of v_0 and v_0 predecessor of v_k if $v_i \in V$ and $v_{i+1} \in S(v_i)$ for all $i \in \{0, \ldots, k-1\}$;*
4. *$T := (V, S)$ tree (with root $\operatorname{root}(T) := r \in V$ and leaves $\mathcal{L}(T) := \{v \in V : S(v) = \emptyset\}$) if*
   - *$\cup_{v \in V} S(v) = V \setminus \{r\}$ and*
   - *each $v \in V \setminus \{r\}$ has exactly one father and is a successor of r;*
5. *the length of the (unique) path from r to v the level-number $\operatorname{level}(v)$ (of v);*
6. *$\operatorname{depth} := \max\{\operatorname{level}(v) : v \in V\}$ the depth of the tree T;*
7. *$T' := (V', S')$ a subtree of T (or $T' \subseteq T$) if T' is a tree with $V' \subseteq V$ and $S'(v) \subseteq S(v)$ for all $v \in V'$.*

Definition 3.8. *Let I be a set, T a tree and $\mu\colon T \to \mathcal{P}(I) \setminus \{\emptyset\}$ a labeling mapping. T is a set decomposition tree corresponding to I if these conditions hold:*

1. *$\mu(\operatorname{root}(T)) = I$,*
2. *$\mu(s) \cap \mu(s') = \emptyset$ for all $v \in T$ and for all $s, s' \in S(v)$ with $s \neq s'$,*
3. *$\dot\cup_{s \in S(v)} \mu(s) = \mu(v)$ for all $v \in T \setminus \mathcal{L}(T)$.*

A set decomposition tree $T(I)$ corresponding to I is called cluster tree if the following conditions are met:

1. *I is a (finite) index set with $\operatorname{root}(T(I)) = I$,*
2. *$\dot\cup_{\sigma \in S(\tau)} \sigma = \tau$ for all $\tau \in T(I) \setminus \mathcal{L}(T(I))$,*
3. *$T(I) \subseteq \mathcal{P}(I) \setminus \{\emptyset\}$.*

Hence, (if $S(\tau) \neq 1$ holds for all $\tau \in T(I)$) we can identify a vertex of a cluster tree with its label and call

1. $\tau,\ \sigma \in T(I)$ *cluster,*
2. *a cluster tree* $T(I \times J)$ *for the index set* $I \times J$ *block cluster tree,*
3. $b = \tau \times \sigma \in T(I \times J)$ *block,*
4. $P \subseteq T(I \times J)$ *partition (of* $I \times J$*) if* $\dot\cup_{b \in P} b = I \times J$ *holds.*

Definition 3.9. *Let* I, J *be finite index sets with* $I \neq \emptyset \neq J$*, cluster trees* $T(I)$*,* $T(J)$ *and a block cluster tree* $T(I \times J)$ *with partition* $P \subseteq T(I \times J)$*. Let* $r \colon P \to \mathbb{N}_0$ *be a local rank distribution and* $n_{\min} \in \mathbb{N}$*. We define the following sparsity constants*

1. $C_{\mathrm{sp,l}}(\tau, P) := \#\{\sigma \in T(J) : \tau \times \sigma \in P\}$ *for* $\tau \in T(I)$*;*
2. $C_{\mathrm{sp,r}}(\sigma, P) := \#\{\tau \in T(I) : \tau \times \sigma \in P\}$ *for* $\sigma \in T(J)$*;*
3. $C_{\mathrm{sp}}(P) := \max\left\{\max_{\tau \in T(I)} C_{\mathrm{sp,l}}(\tau, P),\ \max_{\sigma \in T(J)} C_{\mathrm{sp,r}}(\sigma, P)\right\}$*;*

and boolean functions

1. $\mathrm{size}_{T(I)} \colon \mathcal{P}(I) \to \{\mathrm{true}, \mathrm{false}\},\ \tau \mapsto (\#\tau > n_{\min})$*;*
2. $\mathrm{size}_{T(I\times J)} \colon \mathcal{P}(I) \times \mathcal{P}(J) \to \{\mathrm{true}, \mathrm{false}\},\ (\tau, \sigma) \mapsto (\min\{\#\tau, \#\sigma\} > n_{\min})$*;*
3. $\mathrm{size}_{T(I\times J)}(b) := \mathrm{size}_{T(I\times J)}(\tau, \sigma)$*;*

and call

1. *a block* $b' = \tau' \times \sigma'$ *with* $\tau' \subseteq \tau \subseteq I$*,* $\sigma' \subseteq \sigma \subseteq J$ *a subblock (of the block* $b = \tau \times \sigma$*) and write* $b' \subseteq b$*;*
2. *a boolean function* adm *of the form* $\mathrm{adm} \colon \mathcal{P}(I) \times \mathcal{P}(J) \to \{\mathrm{true}, \mathrm{false}\}$ *(or* $\mathrm{adm}(b) := \mathrm{adm}(\tau, \sigma)$ *for* $b = \tau \times \sigma$*) an admissibility condition if for all* $\tau \subseteq I, \sigma \subseteq J$ *with* $\mathrm{adm}(\tau, \sigma) = \mathrm{true}$ *it follows that* $\mathrm{adm}(\tau', \sigma') = \mathrm{true}$ *holds for all subblocks* $b' = \tau' \times \sigma' \subseteq b$*;*
3. *a partition* P *admissible if either* $\mathrm{adm}(b) = \mathrm{true}$ *or* $\mathrm{size}_{T(I\times J)}(b) = \mathrm{false}$ *for all* $b \in P$*.*

For an admissible partition P *we define the near-field* P^- *and the far-field* P^+ *via*

$$P^- := \{b \in P : \mathrm{size}_{T(I\times J)}(b) = \mathrm{false}\} \quad \text{and} \quad P^+ := P \setminus P^-. \tag{3.3}$$

An $\mathcal{H}$*-matrix or hierarchical matrix (w. r. t. a partition* P *and rank distribution* r*) is a matrix* $M \in \mathbb{R}^{I\times J}$ *with* $M|_b \in \mathcal{R}(r(b), b)$ *for all* $b \in P^+$ *and* $M|_b \in \mathcal{F}(b)$ *for all* $b \in P^-$*. The set of all* $\mathcal{H}$*-matrices w. r. t.* P *and* r *is denoted by* $\mathcal{H}(r, P)$*.*

Definition 3.10. *Let* I, J *be index sets and* $\tau \subseteq I, \sigma \subseteq J$ *clusters with associated sets* $X_i, Y_j \subset \mathbb{R}^d$ *for all* $i \in \tau, j \in \sigma$*. We define*

1. $X_\tau := \cup_{i\in\tau} X_i$ *and* $Y_\sigma := \cup_{j\in\sigma} Y_j$ *as the supports of* τ *and* σ*, resp.;*
2. $\mathrm{diam}(\tau) := \max\{\|x - x'\|_2 : x, x' \in X_\tau\}$ *as the diameter of* τ *and analogously the diameter of* σ*;*
3. $\mathrm{dist}(\tau, \sigma) := \min\{\|x - y\|_2 : x \in X_\tau, y \in Y_\sigma\}$ *as the distance between* τ *and* σ*;*

4. *the smallest axis-parallel cuboid that contains a support of a cluster as its bounding box;*

and call a block $\tau \times \sigma$

1. η*-admissible if* $\min\{\operatorname{diam}(\tau), \operatorname{diam}(\sigma)\} \leq \eta \operatorname{dist}(\tau, \sigma)$ *holds for some* $\eta > 0$*;*
2. *weakly admissible if* $\operatorname{dist}(\tau, \sigma) > 0$ *holds.*

A general definition of an admissibility condition is given in Definition 3.9. Two often used admissibility conditions (namely, the η-admissibility condition and the weak admissibility condition) are introduced in Definition 3.10. In particular, this means that a subblock $b' \subseteq b$ of an η- or weak admissible block b is itself an η- or weak admissible block, resp. Furthermore, their efficient evaluation is discussed in Remark 3.11.

Remark 3.11. *The support* X_τ *of a cluster* τ *is defined in Definition 3.10. However, the evaluation of an admissibility condition such as the* η*-admissibility condition or the weak admissibility condition is typically based on the bounding boxes of the involved clusters* τ *and* σ *(and not on their supports* X_τ *and* X_σ*). The advantage of bounding boxes over supports is that bounding boxes are axis-parallel cuboids such that the evaluation of both types of admissibility conditions can be conducted with low computational cost.*

Theorem 3.12 describes how a low-rank approximation can be computed, which error bounds hold and how to adaptively determine the rank for a given relative tolerance. However, a reduced SVD should be utilized instead of Equation (3.4) to speed up the computation (see Remark 3.13). This theorem is the foundation for the truncation (see Definition 3.14) of matrices $M \in \mathcal{R}(r)$ during computations with $\mathcal{H}$-matrices.

Theorem 3.12. *Let* $M \in \mathbb{R}^{I \times J}$ *with singular value decomposition (SVD)* $M = U\Sigma V^T$ *with singular values* $\sigma_i = \Sigma_{ii}$ *in decreasing order. The matrix*

$$R := U\Sigma_r V^T \text{ with } (\Sigma_r)_{ij} := \begin{cases} \sigma_i, & \text{if } i = j \leq \min\{r, \#I, \#J\}, \\ 0, & \text{else,} \end{cases} \tag{3.4}$$

solves the minimization problems

$$\min_{\operatorname{rank}(R) \leq r} \|M - R\|_2 \quad \text{and} \quad \min_{\operatorname{rank}(R) \leq r} \|M - R\|_F,$$

and the absolute errors are

$$\|M - R\|_2 = \begin{cases} \sigma_{r+1}, & \text{if } r < \min\{\#I, \#J\}, \\ 0, & \text{else,} \end{cases} \quad \text{and} \quad \|M - R\|_F = \sqrt{\sum_{i=r+1}^{\min\{\#I, \#J\}} \sigma_i^2}.$$

Relative errors $\|M - R\|_2 \leq \varepsilon \|M\|_2$ *and* $\|M - R\|_F \leq \varepsilon \|M\|_F$ *with* $\varepsilon > 0$ *are fulfilled for*

$$r \geq r_2(\varepsilon) := \min\{r \in \mathbb{N}_0 : \sigma_{r+1} \leq \varepsilon \sigma_1\} \tag{3.5}$$

and

$$r \geq r_F(\varepsilon) := \min\left\{ r \in \mathbb{N}_0 : \sum_{i=r+1}^{\min\{\#I,\#J\}} \sigma_i^2 \leq \varepsilon^2 \sum_{i=1}^{\min\{\#I,\#J\}} \sigma_i^2 \right\}, \tag{3.6}$$

resp.

Remark 3.13. *The computation of R (3.4) can be accelerated by using a reduced SVD (i. e., neglecting the entries of U, Σ_r and V that are not needed to determine R for the given r) instead of Equation (3.4). Furthermore, $r_2(\varepsilon) = 0 = r_F(\varepsilon)$ holds for all $\varepsilon \geq 1$ in (3.5) and (3.6). In particular, $R = 0$ (i. e., the zero matrix) satisfies a relative error of $\varepsilon \geq 1$.*

Definition 3.14. *Let $r, s \in \mathbb{N}_0$ and $M \in \mathcal{R}(s)$. We call*

$$\mathcal{T}^{\mathcal{R}}_{r \leftarrow s} \colon \mathcal{R}(s) \to \mathcal{R}(r),\ M \mapsto M' = \begin{cases} R \text{ as in (3.4)}, & \text{if } s > r, \\ M, & \text{else,} \end{cases}$$

truncation (to rank r) and define $\mathcal{T}^{\mathcal{R}}_r(M) := \mathcal{T}^{\mathcal{R}}_{r \leftarrow \operatorname{rank}(M)}(M)$. Furthermore, we write $\mathcal{T}^{\mathcal{R}}(M)$ if we do not emphasize the involved ranks r and s. For $\varepsilon > 0$, we set

$$\mathcal{T}^{\mathcal{R}}_{2,\varepsilon}(M) := \mathcal{T}^{\mathcal{R}}_{r_2(\varepsilon) \leftarrow \operatorname{rank}(M)}(M) \quad \text{and} \quad \mathcal{T}^{\mathcal{R}}_{F,\varepsilon}(M) := \mathcal{T}^{\mathcal{R}}_{r_F(\varepsilon) \leftarrow \operatorname{rank}(M)}(M). \tag{3.7}$$

A general definition of a coarsened $\mathcal{H}$-matrix, the underlying coarsened partition and the corresponding rank distribution are provided in Definition 3.15. Matrix blocks which are present in both the original and in the coarsened $\mathcal{H}$-matrix have the same rank (3.8). Examples of coarsened $\mathcal{H}$-matrices are illustrated in the right column of Figure 3.4, whereas the left column of Figure 3.4 shows the corresponding $\mathcal{H}$-matrices before coarsening.

Definition 3.15. *Let $T(I \times J)$ be a block cluster tree, $P := \mathcal{L}(T(I \times J))$ an admissible partition, $r \colon P \to \mathbb{N}_0$ a local rank distribution and $M \in \mathcal{H}(r, P) \subset \mathbb{R}^{I \times J}$ an $\mathcal{H}$-matrix. Let $T'(I \times J)$ be a subtree of $T(I \times J)$ such that for some blocks $b \in T(I \times J)$ all successors of b are deleted (and the son mapping is changed accordingly). We call $P' := \mathcal{L}(T'(I \times J))$ a coarsened partition and $M' \in \mathcal{H}(r', P') \subset \mathbb{R}^{I \times J}$ with $r' \colon P' \to \mathbb{N}_0$ a coarsened $\mathcal{H}$-matrix, if for all $b \in P \cap P'$ holds*

$$r'(b) = r(b). \tag{3.8}$$

Definition 3.16. *Let $M \in \mathbb{R}^{I \times J}$ be a sparse matrix. We call an ordering of M with permutation matrices P_r, P_c such that*

$$P_r M P_c = \begin{pmatrix} M_{D_1} & 0 & M_{D_1 \times S} \\ 0 & M_{D_2} & M_{D_2 \times S} \\ M_{S \times D_1} & M_{S \times D_2} & M_S \end{pmatrix} \tag{3.9}$$

a dissection ordering. It is called nested dissection ordering if M_{D_1} and M_{D_2} are in a (nested) dissection ordering.

Remark 3.17. *The LU factorization of the (nested) dissection ordering* (3.9) *is given by*

$$\begin{pmatrix} M_{D_1} & 0 & M_{D_1\times S} \\ 0 & M_{D_2} & M_{D_2\times S} \\ M_{S\times D_1} & M_{S\times D_2} & M_S \end{pmatrix} = \begin{pmatrix} L_{D_1} & 0 & 0 \\ 0 & L_{D_2} & 0 \\ L_{S\times D_1} & L_{S\times D_2} & L_S \end{pmatrix} \begin{pmatrix} U_{D_1} & 0 & U_{D_1\times S} \\ 0 & U_{D_2} & U_{D_2\times S} \\ 0 & 0 & U_S \end{pmatrix},$$

i. e., the LU factors inherit the zero blocks. Hence, (ideally) the submatrices M_{D_1} and M_{D_2} should be of similar size and large, whereas the other nonzero submatrices should be small. A nested dissection ordering can be obtained via domain decomposition where D_1 and D_2 are disjoint domains and S is a separator (i. e., an interior boundary).

Remark 3.18 summarizes several theoretical results about cost of $\mathcal{H}$-matrix operations [53, Lemmata 6.11, 6.13, 7.17, 7.37 and 7.39, Remarks 5.21 and 6.12, Equation (7.45) and Subsubsection 7.8.5.2].

Remark 3.18. *Let $M \in \mathcal{H}(r,P) \subset \mathbb{R}^{I\times I}$ be an $\mathcal{H}$-matrix with $n_{\min}$ and sparsity constant $C_{\mathrm{sp}}(P)$ as in Definition 3.9, cluster tree $T(I)$, block cluster tree $T(I\times I)$ and $\mathcal{H}$-LU factors $L, U \in \mathcal{H}(r,P) \subset \mathbb{R}^{I\times I}$. It holds that*

- *the cost $S_{\mathcal{H}}(M)$ and $S_{\mathrm{LU}}(M)$ to store M and its $\mathcal{H}$-LU decomposition, resp., is bounded by*

$$\max\{S_{\mathcal{H}}(M), S_{\mathrm{LU}}(M)\} \leq 2 \cdot C_{\mathrm{sp}}(P) \cdot \max\{n_{\min}, r\} \cdot (\mathrm{depth}(T(I)) + 1) \cdot \#I;$$

- *the numbers $N_{\mathrm{MV}}(M)$ and $N_{\mathrm{LU}}(M)$ of arithmetic operations to compute an $\mathcal{H}$-matrix-vector multiplication with M and to apply an $\mathcal{H}$-LU preconditioner of M (i. e., to solve a linear system via forward and backward substitutions), resp., are bounded by*

$$\max\{N_{\mathrm{MV}}(M), N_{\mathrm{LU}}(M)\} \leq 4 \cdot C_{\mathrm{sp}}(P) \cdot \max\{n_{\min}, r\} \cdot (\mathrm{depth}(T(I)) + 1) \cdot \#I;$$

- *the number $N_{\mathrm{LUsetup}}(M)$ of arithmetic operations to compute the $\mathcal{H}$-LU decomposition of M is bounded by*

$$\begin{aligned} N_{\mathrm{LUsetup}}(M) \leq\ & 56 \cdot (C_{\mathrm{sp}}(P))^3 \cdot \max\{n_{\min}, r\} \cdot (\mathrm{depth}(T(I\times I)) + 1)^2 \cdot \#I \\ & + 184 \cdot C_{\mathrm{sp}}(P) \cdot (\max\{n_{\min}, r\})^3 \cdot (\mathrm{depth}(T(I\times I)) + 1) \cdot \#P; \end{aligned}$$

- *the number of blocks in P is bounded by $\#P \leq (2\cdot\#I - 1)\cdot C_{\mathrm{sp}}(P)$. This bound can be tightened to*

$$\#P \leq \left(\frac{4}{n_{\min}} \cdot \#I - 1\right) \cdot C_{\mathrm{sp}}(P) \tag{3.10}$$

if the number of indices in each leaf $\tau \in \mathcal{L}(T(I))$ satisfies $\#\tau \in [0.5\cdot n_{\min}, n_{\min}]$.

We use the notation $n = \#I$ *and assume that* $\operatorname{depth} T(I)$ *and* $\operatorname{depth} T(I \times I)$ *have an asymptotic complexity of at most* $\mathcal{O}(\log n)$ *(which holds for example for a cardinality-based construction of* $T(I)$ *such that clusters* $\tau, \sigma \in T(I)$ *with* $\operatorname{level}(\tau) = \operatorname{level}(\sigma)$ *fulfill* $\|\#\tau - \#\sigma\| \leq 1$*). Furthermore, we assume that* $n_{\min}, r$ *and* $C_{\text{sp}}(P)$ *are independent of* n *(i. e., the number of blocks in* P *has an asymptotic complexity of at most* $\mathcal{O}(n)$*). The above bounds on the cost combined with these assumptions lead to the following bounds on the corresponding asymptotic complexities.*

- *The cost* $S_{\mathcal{H}}(M)$ *and* $S_{\text{LU}}(M)$ *to store an* $\mathcal{H}$*-matrix and its* $\mathcal{H}$*-LU decomposition, resp., and with that the numbers* $N_{\text{MV}}(M)$ *and* $N_{\text{LU}}(M)$ *of arithmetic operations to compute an* $\mathcal{H}$*-matrix-vector multiplication and to apply an* $\mathcal{H}$*-LU preconditioner, resp., have an asymptotic complexity of* $\mathcal{O}(n \log(n))$*.*
- *The number* $N_{\text{LUsetup}}(M)$ *of arithmetic operations to setup an* $\mathcal{H}$*-LU decomposition has an asymptotic complexity of* $\mathcal{O}(n \log^2(n))$*.*

3.4.2 Hierarchical matrices for sparse matrices

We are not aware of any literature about $\mathcal{H}$-matrices in the context of RBF-FD besides [75]. Hence, we conduct a short review of the application of $\mathcal{H}$-matrices in the solution of sparse linear systems of equations that originate from the FE or FD method to illustrate what presumably holds as well for RBF-FD sparse matrices. The main results and observations (that hold for a wide range of applications) are:

- $\mathcal{H}$-matrices can be used to construct direct solvers (the tolerance ε defines then the accuracy [53]) but they are often more efficient as preconditioners for iterative solvers [49];
- $\mathcal{H}$-matrix preconditioners are superior to direct solvers for large problems in $d = 3$ spatial dimensions whereas a direct solver should be preferred for smaller problems or $d \in \{1, 2\}$ [49];
- $\mathcal{H}$-matrix preconditioners are often used in combination with BiCGstab [72, 73, 65] or GMRES [49];
- an $\mathcal{H}$-LU decomposition is significantly faster to compute and more accurate than an $\mathcal{H}$-inverse [72, 65];
- an $\mathcal{H}$-LU decomposition can lead to significantly lower memory cost than the exact LU factors [73];
- the use of nested dissection instead of bisection significantly accelerates the computation of an $\mathcal{H}$-LU decomposition [50, 51, 72, 65];
- $\mathcal{H}$-matrix preconditioners with block cluster trees that are based on nested dissection or bisection lead to similar convergence properties [65];
- the numbers of iterations of the iterative solver increase for larger tolerances ε with the problem size N, whereas (almost) problem size independent numbers of

iterations can be achieved for smaller tolerances ε [73, 65];

- the setup time of an $\mathcal{H}$-matrix preconditioner increases with decreasing tolerance ε and is significantly larger than the solution time of the linear system of equations [72, 73, 65];
- $\mathcal{H}$-matrix preconditioners are especially useful if the linear system should be solved with several right hand sides or if no alternative reliable iterative solver is available (e. g., for highly ill-conditioned problems) [72, 73, 65].

The memory and computation cost can be significantly decreased by using

- geometric instead of blackbox (or algebraic) $\mathcal{H}$-matrix setups [51];
- adaptively determined ranks instead of a fixed rank [73];
- a size function (see Definition 3.9) with

$$n_{\min} \in [10, 100] \tag{3.11}$$

  to avoid too small and too large full matrix blocks [50, 49, 73, 65];
- a weak admissibility condition instead of an η-admissibility condition [50, 54];
- rank-r matrix approximations of the near-field matrix blocks instead of full matrix representations [73].

Further observations are:

- the tolerance ε is chosen in [49, 14, 73, 51, 65, 50] such that it fulfills

$$\varepsilon \in [9 \cdot 10^{-10}, 0.25] \tag{3.12}$$

  whereas $\varepsilon < 9 \cdot 10^{-7}$ is only used for $d = 2$ [50] and $\varepsilon \approx 10^{-4}$ is frequently utilized;
- different norms (such as the Euclidean norm (3.5) and the Frobenius norm (3.6)) can be used to adaptively determine the ranks r [51, 14, 73] (e. g., the Euclidean norm is used in [73], whereas [51, 14, 65] do not state the norm used).

3.4.3 Hierarchical matrices for RBF-FD sparse matrices

The stiffness matrix in the RBF-FD approach is a sparse matrix (i. e., linear memory cost of $\mathcal{O}(N_I)$) and saving it exactly in the $\mathcal{H}$-matrix format (as in the left column of Figure 3.4) would increase the memory cost. This occurs since zero entries in the near-field matrix blocks have to be saved as well, whereas they are usually missing in a sparse matrix format (and additional overhead for the $\mathcal{H}$-matrix format is needed). Nevertheless it can be reasonable to represent a sparse matrix in an $\mathcal{H}$-matrix format (see Subsection 3.4.2). On one hand, an $\mathcal{H}$-matrix can be coarsened (as in the right column of Figure 3.4, the details of coarsening are discussed in Subsection 3.4.4) if an approximation of the sparse matrix is sufficient. This could (depending mainly on the desired accuracy, i. e., the tolerance in the coarsening step, and the number of nonzero

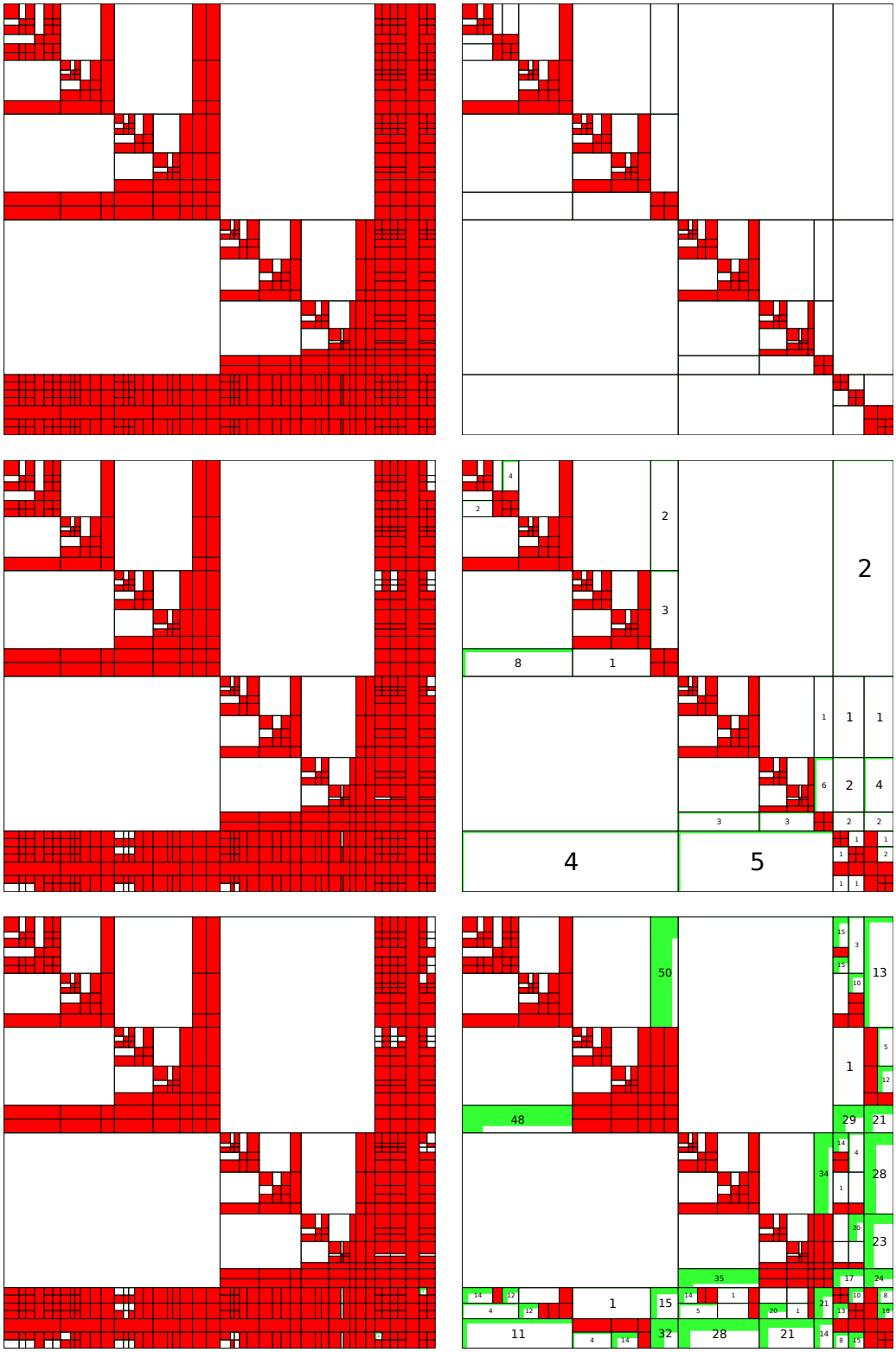

Figure 3.5: $\mathcal{H}$-LU factors corresponding to Figure 3.4 (see Remark 3.19 for setup details).

entries in the sparse matrix) lead to memory savings as well as faster computations compared to a sparse matrix representation of the exact matrix. On the other hand, the $\mathcal{H}$-matrix format allows the computation of an $\mathcal{H}$-LU decomposition, i. e., an LU decomposition with the same block structure (and a typically changed local rank distribution, see Figure 3.5) as the original $\mathcal{H}$-matrix (i. e., low-rank approximations stay low-rank approximations since the corresponding partition P remains unchanged). This can be especially helpful if combined with a nested dissection ordering since the nonzero submatrices of the LU factors of a sparse matrix are typically dense. Furthermore, $\mathcal{H}$-matrices offer a straighforward way (by simply decreasing the tolerance) to construct preconditioners with arbitrarily high accuracies.

Our start setup (derived from the literature review in Subsection 3.4.2) for the solution of the linear system in the RBF-FD approach via an $\mathcal{H}$-matrix ansatz is the following.

- Use nested dissection based $\mathcal{H}$-LU preconditioners for iterative solvers;
- use geometric information about the discretization and the stencils to construct the bounding boxes (see Definition 3.10 and Remark 3.11) and with that the partition of the $\mathcal{H}$-matrix (see Definition 3.9);
- use adaptively determined ranks and follow (3.11).

More details about the construction of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU decomposition are discussed in the following Subsection 3.4.4.

3.4.4 H2Lib

Our numerical tests are performed using the library H2Lib [52]. A straightforward computation of an $\mathcal{H}$-LU preconditioner in this library could utilize the following functions.

1. build_adaptive_dd_cluster to create a cluster tree based on nested dissection,
2. build_nonstrict_block to create a block cluster tree,
3. build_from_block_hmatrix to create an $\mathcal{H}$-matrix structure,
4. copy_sparsematrix_hmatrix to set the entries in the full matrix blocks,
5. coarsen_hmatrix to coarse an $\mathcal{H}$-matrix,
6. lrdecomp_hmatrix to compute an $\mathcal{H}$-LU decomposition.

We discuss their functionality, our choices for the input parameters and our modifications for the following tests.

The index sets of an RBF-FD sparse matrix are $I = J = \{1, \dots, N_I\}$. The associated sets X_i for all $i \in I$ (i. e., for all interior nodes $x_i \in X_\Omega$ (2.1)) are given by the i'th stencil X_i (2.2), i. e.,

$$X_i \quad \text{(associated set in Definition 3.10)} \quad = \quad X_i \quad \text{(stencil in (2.2))}. \tag{3.13}$$

The function build_adaptive_dd_cluster needs geometric information (i. e., the associ-

ated sets X_i for all $i \in I$, see Definition 3.10, or more precisely, a structure _clustergeometry) to determine, for each cluster, the associated bounding box. The structure _clustergeometry contains two $d \times \#I$ arrays. These arrays save the coordinates of the bounding box for each index $i \in I$ (i. e., one array is used for the minimal and one array for the maximal coordinates). In particular, the associated sets X_i (3.13) are not saved but instead only utilized to compute the bounding box for each interior node $x_i \in X_\Omega$ (2.1). These bounding boxes are used consequently to compute the bounding box of each cluster $\tau \in T(I)$ such that an admissibility condition can be efficiently evaluated (see Remark 3.11). Hence, the geometric information as well as the connectivity information about the sparse matrix (i. e., its sparsity structure) are saved in the structure _clustergeometry and used to compute a cluster tree $T(I)$ that represents a nested dissection ordering. The start cluster is the whole index set I, and a cluster is partitioned if it is large enough (w. r. t. the specified $n_{\min}$). A cluster that corresponds to a (row or column) index set of a matrix block M_{D_1} or M_{D_2} (3.9) is partitioned into three sons whereas a cluster that corresponds to a matrix block M_S (3.9) is partitioned only each d'th time (to obtain balanced cluster sizes) into two sons. The direction of the partition is given by the axis-parallel direction with the largest extension of the associated bounding box, and the partition is performed along this direction at the midpoint of the corresponding bounding box.

The cluster tree is in this thesis only computed by the function build_adaptive_dd_cluster. Nonetheless, further more sophisticated algorithms such as an adaptive or alternating construction of the stiffness matrix and its $\mathcal{H}$-matrix representation could be studied. The basic idea of these approaches would be to check after each partitioning step whether the partition can be improved and to update the partition or the stencils (i. e., the sparsity structure of the stiffness matrix) if necessary.

However, two problems can arise if the function build_adaptive_dd_cluster is used for an RBF-FD stiffness matrix. On one hand, the exact representation as an $\mathcal{H}$-matrix can fail (i. e., the computed $\mathcal{H}$-matrix contains zero entries which are actual nonzero entries in the stiffness matrix) if the connectivity information relies on the sparsity structure of the stiffness matrix B (2.14). This happens since the implementation of the function build_adaptive_dd_cluster (implicitly) expects a symmetric sparsity pattern. Therefore, we utilize the symmetrized matrix $B + B^T$ (see Figure 3.6 left side) instead of B for the connectivity information. However, this can lead to additional "false" nonzero entries (i. e., which are actually zero entries, see Figure 3.6 right side). The difference between the numbers of entries in the sparsity patterns of B and $B + B^T$ is (for sufficiently large problem size N) typically below 9 entries per row (below 9% additional entries for all node distributions and all polynomial degrees tested here). Nevertheless, the subsequent block cluster tree can lead to an $\mathcal{H}$-matrix such that there are full matrix blocks that consist solely of zero entries (i. e., which could be represented exactly as rank-0 matrices, see Figure 3.7). Typical reasons for the occurrence of a zero full matrix block are that it consists of at least one "false" nonzero entry or that an η-admissibility condition with a relatively large η value is utilized. Using a weak admissibility condition is not a sufficient condition to guarantee that no zero full matrix

block occurs in an $\mathcal{H}$-matrix. However, a zero full matrix block cannot occur in an $\mathcal{H}$-matrix if the following conditions hold.

- The associated sets of the clusters (and with that the structure _clustergeometry) are given by (3.13).
- A weak admissibility condition is utilized.
- The stiffness B has a symmetric sparsity structure.

Nonetheless, the last condition usually does not hold in our numerical tests and the likelihood of zero full matrix blocks is increased by

- increasing the associated sets of the clusters, i. e., using more nodes than given by (3.13),
- using an η-admissibility condition instead of a weak admissibility condition,
- increasing the η value in the η-admissibility,
- using the symmetrized sparse matrix $B+B^T$ instead of B for the sparsity structure (which can increase the sparsity constants in Definition 3.9 and prevents their a priori determination).

On the other hand, it can occur that one of the three sons which should define a (row or column) index set of a matrix block M_{D_1} or M_{D_2} (3.9) is empty. This means that the combination of the connectivity information about the sparse matrix and the direction of the partition does not lead to a partition of the associated index set into two disjoint nonempty sets (and a separator set, i. e., corresponding to a matrix block M_S (3.9)) such that both resulting matrix blocks M_{D_1} and M_{D_2} (3.9) consist of at least one row and column. An example of this phenomenon is a cluster $\tau \subseteq X_i$ (3.13) (i. e., all indices of the index set τ belong to the same stencil X_i) with $\#\tau > n_{\min}$. Therefore, this index set should be partitioned (since it is larger than $n_{\min}$) but the connectivity information does not allow the partition into two disjoint nonempty sets (and a separator set). Hence, the occurrence of an empty son appears frequently for RBF-FD sparse matrices (in particular, if the stencil size is approximately $n_{\min}$ or larger). Furthermore, the likelihood of this phenomenon is even increased by using the symmetrized matrix $B + B^T$ instead of B. Nonetheless, this phenomenon typically does not occur for FD or FE sparse matrices since these stiffness matrices are usually symmetric (i. e., no additional "false" nonzero entries originate from their symmetrization) and significantly sparser than RBF-FD stiffness matrices (e. g., the stencil sizes in the FD approach in this thesis are 7 and 19, i. e., often significantly smaller than $n_{\min}$ (3.11)). Hence, we stop the partition (in addition to the standard case that the cluster is small according to $n_{\min}$), if an empty son would be generated. However, this can increase the memory cost of an $\mathcal{H}$-matrix (since the upper bound on $S_{\mathcal{H}}(M)$ in Remark 3.18 assumes that $n_{\min}$ is an upper bound on the maximal number of rows and columns of a full matrix block, see Definition 3.9).

The function build_nonstrict_block can use the cluster tree $T(I)$ for the rows and the columns and additionally needs an admissibility condition to create the block cluster tree $T(I \times I)$. The admissibility condition admissible_dd_cluster represents the η-

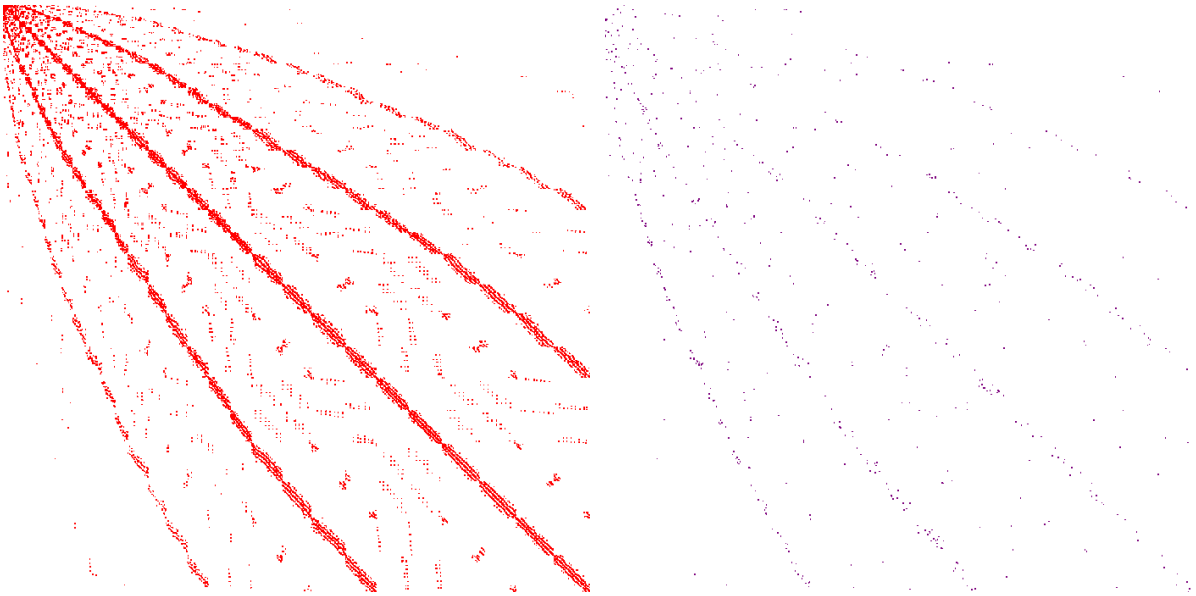

Figure 3.6: Sparsity structure of the symmetrized stiffness matrix $B + B^T$ (left) and the difference between the sparsity structures of $B + B^T$ and B (right) with B as in the left column of Figure 3.1.

admissibility condition enhanced by the knowledge that certain zero blocks in the nested dissection ordering remain zero in the subsequent LU decomposition (i. e., they can be represented exactly as rank-0 matrices, see the first two rows in Figure 3.4). Furthermore, we test a modified version with the weak instead of the η-admissibility condition (see the last row in Figure 3.4). In all numerical tests in this thesis the block cluster tree $T(I \times I)$ is based on a nested dissection ordering. Hence, in the discussion of our numerical results we will utilize only the terms η- and weak admissibility condition (without explicitly stating that they are enhanced by the knowledge that certain zero blocks in the nested dissection ordering remain zero in the subsequent LU decomposition).

The function build_from_block_hmatrix uses the block cluster tree $T(I \times I)$ to create the structure of the $\mathcal{H}$-matrix. The setup so far (i. e., the combination of the associated sets (3.13), the admissibility conditions and the connectivity information used) guarantees that all far-field blocks are zero blocks (similar to a FE sparse matrix) since all nonzero entries of B are saved in full matrix blocks. The function copy_sparsematrix_hmatrix sets the entries in the full matrix blocks and coarsen_hmatrix coarsens the $\mathcal{H}$-matrix.

The function coarsen_hmatrix uses an $\mathcal{H}$-matrix $M \in \mathcal{H}(r, P) \subset \mathbb{R}^{I \times J}$ (see Definition 3.15), a norm and a tolerance ε (see Theorem 3.12) as input parameters to compute a coarsened $\mathcal{H}$-matrix $M' \in \mathcal{H}(r', P') \subset \mathbb{R}^{I \times J}$ (see Definition 3.15). The output $M' = M$ indicates that the $\mathcal{H}$-matrix remains unchanged, whereas $P' = \{I \times J\}$ means that M' is a low-rank matrix. The basic ideas of the function coarsen_hmatrix are described in the following.

- The candidates for coarsening are blocks b (see Definition 3.8) which fulfill

$$b = \sigma \times \tau \in T(I \times J) \quad \text{with} \quad \sigma \neq \tau \quad \text{and} \quad S(b) \subseteq P, \tag{3.14}$$

  i. e., all sons of b are leaves.
- A low-rank representation R_b (see Definition 3.6) of $M|_b$ is created for each b that fulfills (3.14) by agglomerating the low-rank approximations of each son $S(b)$ where a full matrix is represented by its multiplication with an identity matrix of adequate size.
- The truncation $\mathcal{T}^{\mathcal{R}}(R_b)$ of R_b (see Definition 3.14 and Theorem 3.12) and its memory cost are computed.
- The memory cost of the low-rank matrices and full matrices that correspond to the matrix block $M|_b$ is computed.
- The representation $\mathcal{T}^{\mathcal{R}}(R_b)$ is used instead of the matrix block $M|_b$ (i. e., coarsening is done) if the memory cost of $\mathcal{T}^{\mathcal{R}}(R_b)$ is lower than the memory cost of the matrix block $M|_b$.
- The $\mathcal{H}$-matrix structure (i. e., the local rank distribution r and the partition P in (3.14)) are updated after each successful coarsening step.

The successive execution of the functions build_from_block_hmatrix, copy_sparsematrix_hmatrix and coarsen_hmatrix can be problematic since build_from_block_hmatrix directly allocates memory for all near-field matrix blocks and can therefore lead to large peaks in the memory usage (as stated in [52, Issue #36 in the corresponding GitHub repository]). Hence, we followed the recommendations in this issue to implement a function that combines the tasks of these three functions. The basic idea is to postpone the complete memory allocation and instead alternate necessary allocation and filling of the near-field matrix blocks (i. e., only the matrix blocks which correspond to the sons $S(b)$ of a block b that satisfies (3.14) needs to be allocated and filled) with coarsening (i. e., compute R_b, the truncation $\mathcal{T}^{\mathcal{R}}(R_b)$ and its memory cost). Furthermore, the function coarsen_hmatrix ignores that due to Remark 3.13

$$\mathcal{T}^{\mathcal{R}}(R_b) = 0 \quad \text{holds for all blocks } b \text{ if a relative tolerance } \varepsilon \geq 1 \text{ is used.} \tag{3.15}$$

This means that coarsening of an $\mathcal{H}$-matrix with tolerance $\varepsilon \geq 1$ (w. l. o. g., $\varepsilon = 1$) followed by its $\mathcal{H}$-LU decomposition with the same tolerance ε is a special case that combines properties known from the (block-) Jacobi preconditioner and the ILU(0) decomposition. On one hand, after coarsening, all full matrix blocks in the $\mathcal{H}$-matrix are on the diagonal (i. e., as in the case of a (block-) Jacobi preconditioner) or close to the diagonal (see the plot on the right of the top row in Figure 3.4) since (3.15) holds and the memory cost of a rank-0 matrix is always below the memory cost accumulated over all sons $S(b)$ of the individual matrix blocks $M|_{S(b)}$. Hence, coarsening is done for all blocks b satisfying (3.14) and all matrix blocks that are not coarsened are leaves (i. e., their numbers of rows and columns are bounded by $n_{\min}$ (3.11)). Consequently, the

conditions (3.14) could be used directly to define an admissibility condition which represents coarsening with $\varepsilon \geq 1$. On the other hand, all admissible blocks of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU decomposition are represented as rank-0 matrices (i. e., the memory cost of the $\mathcal{H}$-LU decomposition is known a priori and identical to the memory cost of the $\mathcal{H}$-matrix, i. e., as in the case of an ILU(0) decomposition). Therefore, we modified the coarsening functions to take this into account, i. e., to avoid the setup of R_b for the computation of the (a priori known) (3.15) if a tolerance $\varepsilon \geq 1$ is utilized. Additionally, we test modified versions of the copy functions (with and without coarsening), which count the nonzero entries in a full matrix block to detect zero full matrix blocks and convert them to rank-0 blocks.

Figures 3.4, 3.5 and 3.7 illustrate that coarsening as introduced in this subsection is not advisable for a small tolerance ε since the right column of Figure 3.4 reveals that the amount of coarsened matrix blocks decreases with decreasing tolerance ε (i. e., from the top row to the bottom row). The reason for this observation is that the rank of the truncation $\mathcal{T}^{\mathcal{R}}(R_b)$ of R_b increases with decreasing tolerance ε and a higher rank leads to higher memory cost of $\mathcal{T}^{\mathcal{R}}(R_b)$. Hence, the likelihood of using the coarsened representation $\mathcal{T}^{\mathcal{R}}(R_b)$ decreases for decreasing tolerance ε, whereas the computation cost of R_b and its truncation $\mathcal{T}^{\mathcal{R}}(R_b)$ is independent of the tolerance ε. Therefore, the potential of coarsening (w. r. t. its computational cost) decreases with decreasing tolerance ε (and the extreme case would be that the matrices R_b and their truncation $\mathcal{T}^{\mathcal{R}}(R_b)$ are computed without using any of these coarsened matrix blocks). Moreover, this coarsening approach does not prevent small full matrix blocks which could be coarsened such that they are represented as one full matrix block. However, there are further (not studied here) coarsening functions in the H2Lib [52] such as cut_cluster which bounds the depth of the cluster tree and coarsen_cluster which removes too small clusters. These functions can modify the cluster tree, i. e., they are applied before the block cluster tree is computed.

Remark 3.19. *Figure 3.4 illustrates in the left column exact representations of a stiffness matrix and in the right column coarsening with the tolerances* $\varepsilon \in \{1, 0.7, 0.01\}$. *The rows show the* η- *and the weak admissibility conditions as well as different tolerances* ε, *namely*

- $\eta = 3$ *with* $\varepsilon = 1$ *(top row),*
- $\eta = 15$ *with* $\varepsilon = 0.7$ *(middle row) and*
- *weak admissibility condition with* $\varepsilon = 0.01$ *(bottom row).*

The size function (see Definition 3.9) utilizes $n_{\min} = 35$. *The plots in the left column illustrate that the* η*-admissibility condition converges to the weak admissibility condition for* $\eta \to \infty$ *(whereas the weak admissibility condition should be preferred over the* η*-admissibility condition w. r. t. the computational cost). Furthermore, we observed that for this stiffness matrix and a fixed tolerance* $\varepsilon \in \{1, 0.7, 0.01\}$ *the resulting coarsened* $\mathcal{H}$*-matrix is identical irrespective of the admissibility condition used. Hence, we recommend to use a weak admissibility condition if coarsening should be applied afterwards.*

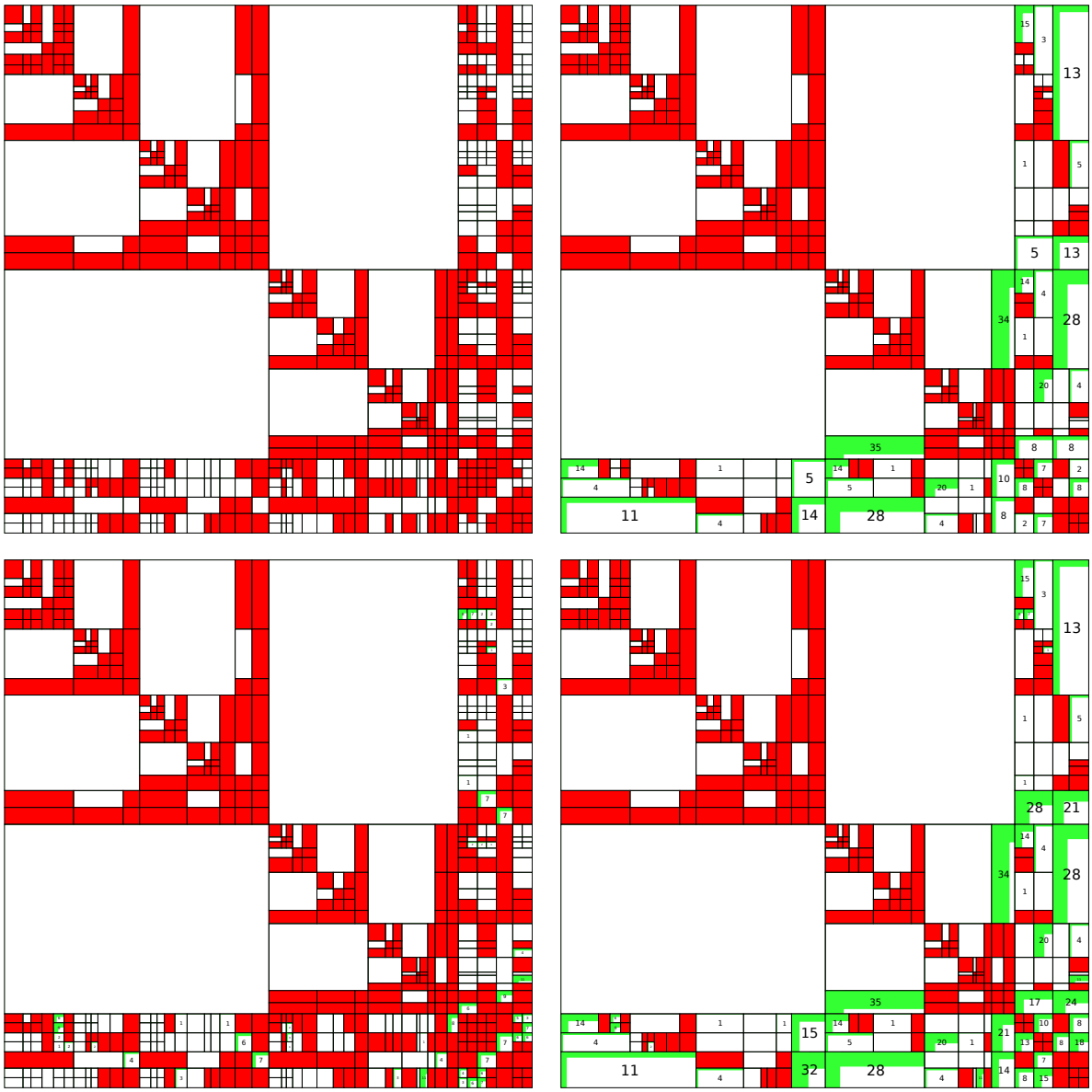

Figure 3.7: $\mathcal{H}$-matrices for the stiffness matrix shown in the left column of Figure 3.1 (top row) and its $\mathcal{H}$-LU factors (bottom row) with weak admissibility condition, $n_{\min} = 35$ and tolerance $\varepsilon = 0.01$ without (left) and with coarsening w. r. t. Euclidean norm (right). Conversion of zero full matrix blocks to rank-0 blocks is used; the top and bottom rows are otherwise identical to the last row in Figures 3.4 and 3.5, resp.

The advantages and disadvantages of the conversion of zero full matrix blocks to rank-0 matrices are that this conversion can reduce the memory cost of the $\mathcal{H}$-matrix and the computation times of its coarsening. However, such a conversion is not necessary if the tolerance $\varepsilon = 1$ is utilized since (3.15) can be used directly. Nevertheless, it can

reduce the computation time of the $\mathcal{H}$-LU decomposition as well as its memory cost (see Figure 3.7). However, on one hand, the memory cost of the $\mathcal{H}$-LU decomposition could be as well increased since a zero full matrix block in the $\mathcal{H}$-matrix may turn into a nonzero block in the $\mathcal{H}$-LU decomposition and the maximal rank r in an $\mathcal{H}$-matrix $M \in \mathcal{H}(r, P)$ influences its memory cost (see Remark 3.18). Hence, the memory cost of the resulting $\mathcal{H}$-LU decomposition can be higher than in the case without conversion of zero full matrix blocks to rank-0 matrices if too large ranks are needed to represent the converted blocks as "low-rank" approximations (since the rank of a matrix block is solely bounded by the minimum of the numbers of rows and columns of this block). Furthermore, decreasing the tolerance ε can increase the necessary rank r in Theorem 3.12, i. e., it is more likely that a "low-rank" approximation in an $\mathcal{H}$-LU decomposition has a too large rank. Moreover, the smaller the tolerance ε is, the less coarsening is done and the more likely are small rank-0 matrix blocks. On the other hand, the combination of the η-admissibility condition (since it leads to more zero full matrix blocks than for example the weak admissibility condition) with the conversion of zero full matrix blocks to rank-0 matrices is suboptimal and wasteful (see Remark 3.19). Furthermore, coarsening an $\mathcal{H}$-matrix in $\mathcal{H}(r, P)$ (see Definition 3.9) as well as the conversion of zero full matrix blocks to rank-0 matrices can be interpreted as a posteriori modifications to

- the partition P (i. e., to the block cluster tree $T(I \times J)$ since its leaves $\mathcal{L}(T(I \times J))$ are the partition P),
- the separation of near-field P^- and far-field P^+ (i. e., to the admissibility condition adm (3.3)),
- the local rank distribution r (since $r\colon P \to \mathbb{N}_0$, i. e., the domain of r is the underlying partition P).

The function lrdecomp_hmatrix computes the $\mathcal{H}$-LU decomposition. This function, as well as the coarsening functions, need a function to measure the errors in some norm (e. g., the relative or absolute errors in the Euclidean or Frobenius norm) and a tolerance to define the truncation (see Definition 3.14) to determine the ranks of the low-rank approximations (see Definition 3.6) in the $\mathcal{H}$-matrix. In the following numerical tests, we utilize these functions for a given test setup solely with the same norm and tolerance. However, the combination of different norms or tolerances is feasible, whereas especially the combination of coarsening the $\mathcal{H}$-matrix with a high tolerance (i. e., the truncation $\mathcal{T}^{\mathcal{R}}(R_b)$ of R_b typically has a relatively low rank) followed by its $\mathcal{H}$-LU decomposition with a small tolerance (i. e., the matrix block corresponding to $\mathcal{T}^{\mathcal{R}}(R_b)$ usually needs a relatively high rank, possibly even with higher cost than a corresponding full matrix) is wasteful and should be avoided since this would lead to a high accuracy approximation of the LU factors of a low accuracy approximation of the stiffness matrix.

3.4.5 Numerical results for a small degree of polynomial augmentation

In this subsection, we examine numerical tests to answer the following questions for discretizations with polynomial degree $\ell = 2$:

- Should a weak admissibility condition (optionally with conversion of zero full matrix blocks to rank-0 blocks) or an η-admissibility condition be preferred?
- Is coarsening of the $\mathcal{H}$-matrix (or the $\mathcal{H}$-LU decomposition) useful?
- How to select $n_{\min}$?
- Which norm and tolerance ε should be used in the truncation?

We solely show results with the LPBiCGstab solver since further tests with LPGMRES and RPGMRES (not included here) indicated that the situation for $\mathcal{H}$-LU preconditioners is similar as for ILU(0) preconditioners. GMRES typically leads to a few more iterations than BiCGstab (as long as BiCGstab converges) and the impact of the method of application of the preconditioner (e. g., left or right) is negligible if the relative residual tolerance in the stopping criterion of the iterative solver is small and the condition number of the stiffness matrix is relatively low (see Figure 2.21). The main advantage of GMRES over BiCGstab is that GMRES even leads to (possibly slow) convergence for poor preconditioners, but GMRES becomes memory and time consuming especially if the number of iterations is large. However, our aim is the construction of good preconditioners which should (ideally) lead to fast convergence for all solvers (i. e., for left, right and split preconditioning [125] for BiCGstab and GMRES).

The results of this subsection are presented through the following figures.

- Figure 3.8: Influence of the problem size N on different grid types (Cartesian, PNP, Halton) for different truncation tolerances ε and PHS($k = 2,\ \ell = 2$) (see Notation 3.1) (2.24) and PHS($k = 3,\ \ell = 2$) (2.26) on the computation time.
- Figure 3.9: Influence of the problem size N on different grid types (Cartesian, PNP, Halton) for different truncation tolerances ε and PHS($k = 2,\ \ell = 2$) (2.24) and PHS($k = 3,\ \ell = 2$) (2.26) on the numbers of iterations and the memory cost.
- Figure 3.10: Relationship between the computation time on different grid types (Cartesian, PNP, Halton) and the truncation tolerances ε and PHS($k = 2,\ \ell = 2$) (2.24) and PHS($k = 3,\ \ell = 2$) (2.26) on the approximation error.
- Figure 3.11: Influence of the problem size N on the computation time for different preconditioner setups with truncation tolerances $\varepsilon \in \{1, 0.1\}$ for PHS($k = 2,\ \ell = 2$) (2.24).
- Figure 3.12: Influence of the problem size N on the numbers of iterations and the memory cost of different preconditioner setups with truncation tolerances $\varepsilon \in \{1, 0.1\}$ for PHS($k = 2,\ \ell = 2$) (2.24).
- Figure 3.13: Influence of the problem size N on the computation time for different

preconditioner setups with truncation tolerances $\varepsilon \in \{1, 0.1\}$ for PHS($k = 2,\ \ell = 2$) (2.24).

- Figure 3.14: Influence of the problem size N on the numbers of iterations and the memory cost of different preconditioner setups with truncation tolerances $\varepsilon \in \{1, 0.1\}$ for PHS($k = 2,\ \ell = 2$) (2.24).

Several of these figures and of the figures in the following Subsection 3.4.6 include an additional color/marker "RBF-FD" which represents either the setup time of the linear system (without the setup time of the node distribution), which is also shown with a different scaling factor (i. e., $1/(N \log(N))$ instead of $1/(N \log^2(N))$) in Figures 3.2 and 3.3, or the memory cost of the stiffness matrix, which is identical to the memory cost of its ILU(0) decomposition.

We start with the statement of our recommended $\mathcal{H}$-LU preconditioner setup in Remark 3.20 and illustrate its performance on different node distributions for the cases with PHS($k = 2,\ \ell = 2$) (2.24) and PHS($k = 3,\ \ell = 2$) (2.26). Afterwards, we focus on PNP nodes, discuss why we would recommend this $\mathcal{H}$-LU preconditioner setup for the solution of the linear system in the RBF-FD approach (with PHS generating function and polynomial augmentation). We demonstrate in addition what can and what should not be changed in this recommended $\mathcal{H}$-LU preconditioner setup.

Remark 3.20. *Our tests (partly included in this subsection) suggest to*

- *use a weak admissibility condition (instead of an η-admissibility condition);*
- *convert zero full matrix blocks in the $\mathcal{H}$-matrix to rank-0 blocks;*
- *coarsen the $\mathcal{H}$-matrix (and avoid coarsening of the $\mathcal{H}$-LU decomposition);*
- *use a tolerance ε (for both the coarsening and the $\mathcal{H}$-LU decomposition, see Subsection 3.4.4) that is close to 1 (i. e., slightly larger than in the literature (3.12), especially if the linear system of equations is relatively small, if it should be solved only for one right hand side or if the PHS degree is determined by (2.24));*
- *use (3.11) with smaller $n_{\min}$ for smaller stencil sizes or higher tolerances ε, and larger $n_{\min}$ for larger stencil sizes or smaller tolerances ε.*

For simplicity (i. e., without conducting comprehensive numerical tests), we mainly use the relative error w. r. t. the Euclidean norm, i. e., we utilize rank-r matrices with $r = r_2(\varepsilon)$ (3.5). Hence, this defines our recommended $\mathcal{H}$-LU preconditioner setup, and the parameters for problem dependent fine-tuning are mainly the tolerance ε (3.5) and $n_{\min}$ (see Definition 3.9). Therefore, in the following, we only state the values used for these two parameters if we mention that our recommended $\mathcal{H}$-LU preconditioner setup is utilized.

Figure 3.8 shows the computation times scaled by $1/(N \log^2(N))$ w. r. t. the number of nodes N for our recommended $\mathcal{H}$-LU preconditioner setup with $n_{\min} = 35$. Each color/marker represents a different tolerance $\varepsilon \in \{1, 0.7, 0.1, 0.01, 0.001\}$. The additional color/marker "RBF-FD" represents the setup time of the linear system (without the

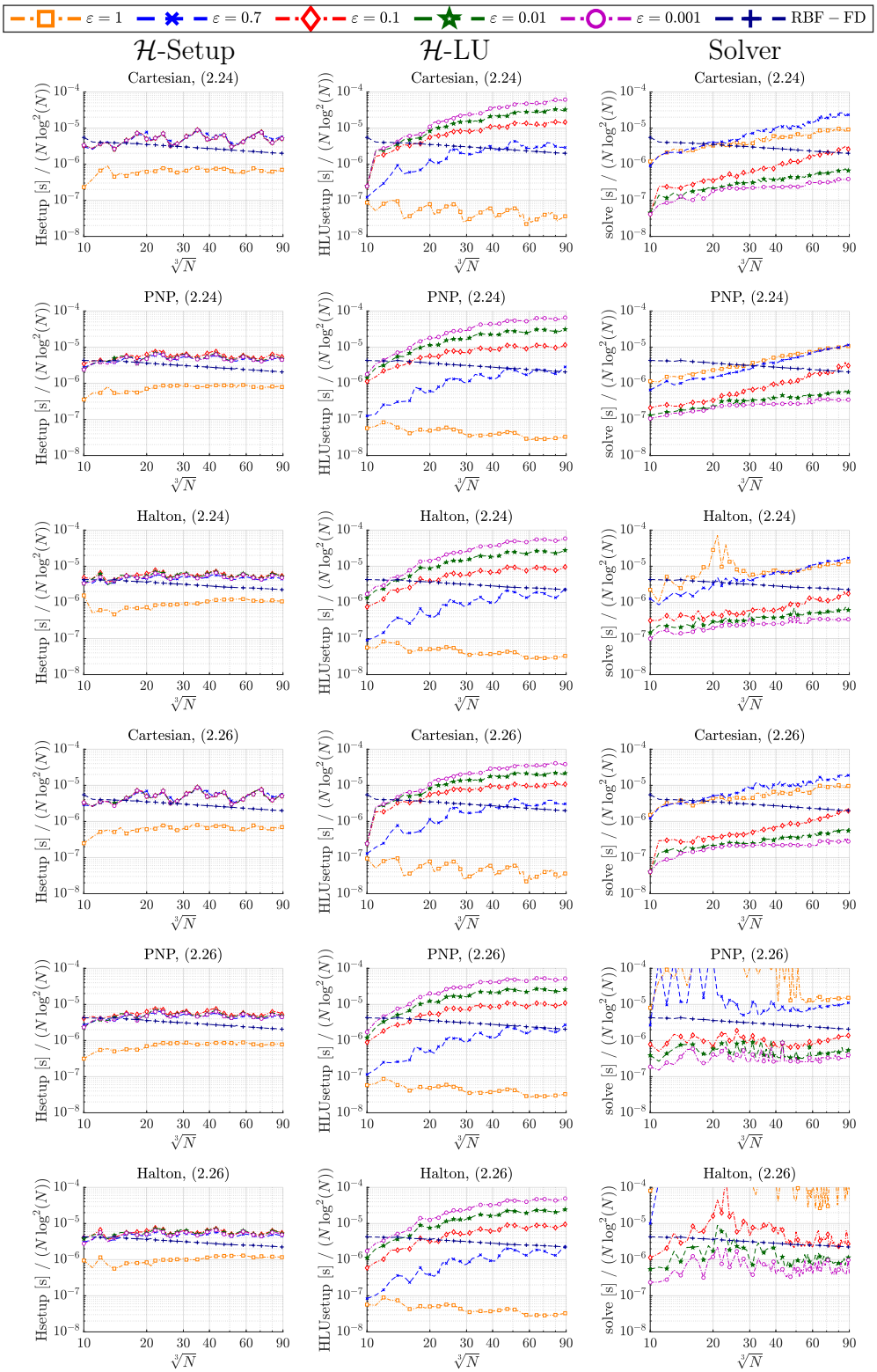

Figure 3.8: Scaled computation times (in seconds) to setup the $\mathcal{H}$-matrix (left), the $\mathcal{H}$-LU decomposition (middle) and to solve the linear system (right) for PHS($k = 2$, $\ell = 2$) (2.24) and PHS($k = 3$, $\ell = 2$) (2.26) and recommended preconditioner setup with $n_{\min} = 35$ and tolerances $\varepsilon \in \{1, 0.7, 0.1, 0.01, 0.001\}$ for several node distributions (Cartesian, PNP, Halton).

setup time of the node distribution) which is the same for all three columns. Each row corresponds to a different combination of node distribution (Cartesian, PNP, Halton) and PHS degree (PHS($k = 2,\ \ell = 2$) (2.24) and PHS($k = 3,\ \ell = 2$) (2.26)) and each column represents a different part of the computation ($\mathcal{H}$-matrix setup, $\mathcal{H}$-LU decomposition, solution of the linear system).

The plots in the left column indicate that the complexity of the $\mathcal{H}$-matrix setup is (according to Remark 3.3) approximately $\mathcal{O}(N \log^2(N))$ (i. e., larger than the complexity of the setup of the linear system). The truncation tolerance ε controls the accuracy of the coarsened $\mathcal{H}$-matrix and has a negligible influence on the setup time for $\varepsilon < 1$, whereas the speedup for $\varepsilon = 1$ is gained by using (3.15). The results are qualitatively similar for all tested node distributions and both PHS degrees.

The middle column illustrates that the truncation tolerance ε has a large influence on the computation time of the $\mathcal{H}$-LU decomposition. An increase in ε leads to a decrease in the computation time. The complexity of the $\mathcal{H}$-LU decomposition lies below $\mathcal{O}(N \log^2(N))$ for $\varepsilon = 1$ and above $\mathcal{O}(N \log^2(N))$ for $\varepsilon < 1$ (i. e., approximately $\mathcal{O}(N \log^4(N))$). These results indicate that for tolerances $\varepsilon < 1$ the complexity of the $\mathcal{H}$-LU decomposition is above $\mathcal{O}(N_I \log^2(N_I))$, i. e., the assumptions in Remark 3.18 do not hold. Further tests (not included here) for $\mathcal{H}$-LU decompositions of 7-point FD stiffness matrices revealed that the reason for the increase in the asymptotic complexity for tolerances $\varepsilon < 1$ lies in the coarsening, i. e., the asymptotic complexity is $\mathcal{O}(N_I \log^2(N_I))$ for setups without coarsening and above $\mathcal{O}(N_I \log^2(N_I))$ for setups with coarsening. Moreover, coarsening does not change the depth of the (block) cluster tree and $n_{\min}$ is fixed in this comparison. Hence, these tests demonstrate that the maximal rank r or the sparsity constant $C_{\text{sp}}(P)$ can increase with N if coarsening is used. The results are, again, qualitatively similar for all tested node distributions and both PHS degrees.

The right column demonstrates that an increase in the tolerance ε typically leads to an increase in the solution time (except for the comparison between $\varepsilon = 1$ and $\varepsilon = 0.7$). The complexity of the solution of the linear system for smaller ε values lies above $\mathcal{O}(N \log^2(N))$ (i. e., approximately $\mathcal{O}(N \log^4(N))$), whereas larger tolerances ε can lead to even higher complexities. The solution time for PHS($k = 2,\ \ell = 2$) (2.24) on Halton nodes with $\varepsilon = 1$ is drastically increased for some N values, but otherwise the results are qualitatively similar for all tested node distributions. The solution time on Cartesian nodes is almost identical for both PHS degrees. However, there are large oscillations in the solution times for PHS($k = 3,\ \ell = 2$) (2.26) on PNP and Halton nodes and the solution times can drastically increase for the higher tolerances $\varepsilon \geq 0.1$.

Taking all plots into account, we see that there is a tradeoff between the solution time and the setup time of the preconditioner. A lower ε leads to a higher accuracy of the preconditioner and therefore typically to a faster solution of the linear system in exchange for a longer setup time, and vice versa. The computation time of the overall solution approach for discretizations with larger N is dominated by the $\mathcal{H}$-LU decomposition (for smaller tolerances ε) or by the solution of the linear system (for larger ε). A higher tolerance (especially if $\varepsilon = 1$) can increase the complexity of the solution and decrease the

complexity of the preconditioner setup, and vice versa. However, a too large tolerance ε can significantly increase the solution time (especially for PHS($k = 3,\ \ell = 2$) (2.26)).

Figure 3.9 shows the numbers of iterations and the memory cost of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU factors both scaled by $1/(N \log^2(N))$ for the same setups as in Figure 3.8. The additional color/marker "RBF-FD" is only used in the middle and right column (in particular, it is for a fixed row the same curve in both columns) and represents the memory cost of the stiffness matrix. The left column shows that the numbers of iterations increase with the problem size N for larger tolerances ε whereas lower tolerances ε can lead to (almost) constant numbers of iterations. The numbers of iterations for Halton nodes with $\varepsilon = 1$ drastically increase for some N values (which explains the drastically increased solution times in Figure 3.8) but otherwise the results are qualitatively similar for all tested node distributions.

The middle column illustrates that for all tolerances ε the memory cost of the $\mathcal{H}$-matrix is higher than for the stiffness matrix (which has the same memory cost as the ILU(0) preconditioner). Higher tolerances lead to lower memory cost, and vice versa. The differences in the memory cost decrease for decreasing tolerance ε and are almost invisible for $\varepsilon \leq 0.01$ (since the limit $\varepsilon \to 0$ corresponds to the case without coarsening). However, the scaled memory cost oscillates with the problem size N (especially on Cartesian nodes with $\varepsilon = 1$) and their asymptotic complexity lies between $\mathcal{O}(N \log(N))$ and $\mathcal{O}(N \log^2(N))$ for $\varepsilon < 1$ and is smaller than $\mathcal{O}(N \log^2(N))$ for $\varepsilon = 1$. The memory cost of the stiffness matrix is $\mathcal{O}(N)$. The results are qualitatively similar for PHS($k = 2,\ \ell = 2$) (2.24) and PHS($k = 3,\ \ell = 2$) (2.26) and for all tested node distributions except that for Cartesian nodes the oscillations in the memory cost are larger and the differences between the tolerances are smaller for $\varepsilon \leq 0.1$. These especially visible oscillations on Cartesian nodes for tolerance $\varepsilon = 1$ can contribute to the oscillations in the following $\mathcal{H}$-LU decomposition (see Figure 3.8 for its computation time, and the right column of Figure 3.9 for its memory cost). Hence, it could be further studied whether the oscillations in the amount or size of the blocks that do not satisfy (3.14) are larger for Cartesian nodes than for PNP or Halton nodes.

The right column shows that many of the observations regarding the memory cost of the $\mathcal{H}$-matrix also hold for its $\mathcal{H}$-LU factors. The main differences are that for smaller tolerances a change in the tolerance ε leads to significantly larger changes in the memory cost of the $\mathcal{H}$-LU factors than for the $\mathcal{H}$-matrix and the oscillations in the scaled memory cost are reduced for smaller tolerances $\varepsilon \leq 0.1$. These less visible oscillations for smaller tolerances $\varepsilon \leq 0.1$ can originate from the relatively large memory cost of the $\mathcal{H}$-LU factors (e. g., for a small tolerance ε the difference between the memory cost of an $\mathcal{H}$-matrix and its $\mathcal{H}$-LU factors can be larger by more than a factor of two). The memory cost of the $\mathcal{H}$-LU factors can be combined with the numbers of iterations to explain the solution times in Figure 3.8 (since higher memory cost leads to higher computation cost per iteration). The results are qualitatively similar for all tested node distributions and for PHS($k = 2,\ \ell = 2$) (2.24) and PHS($k = 3,\ \ell = 2$) (2.26). Similarly for the asymptotic complexities of the computation times, the asymptotic complexities of the memory cost

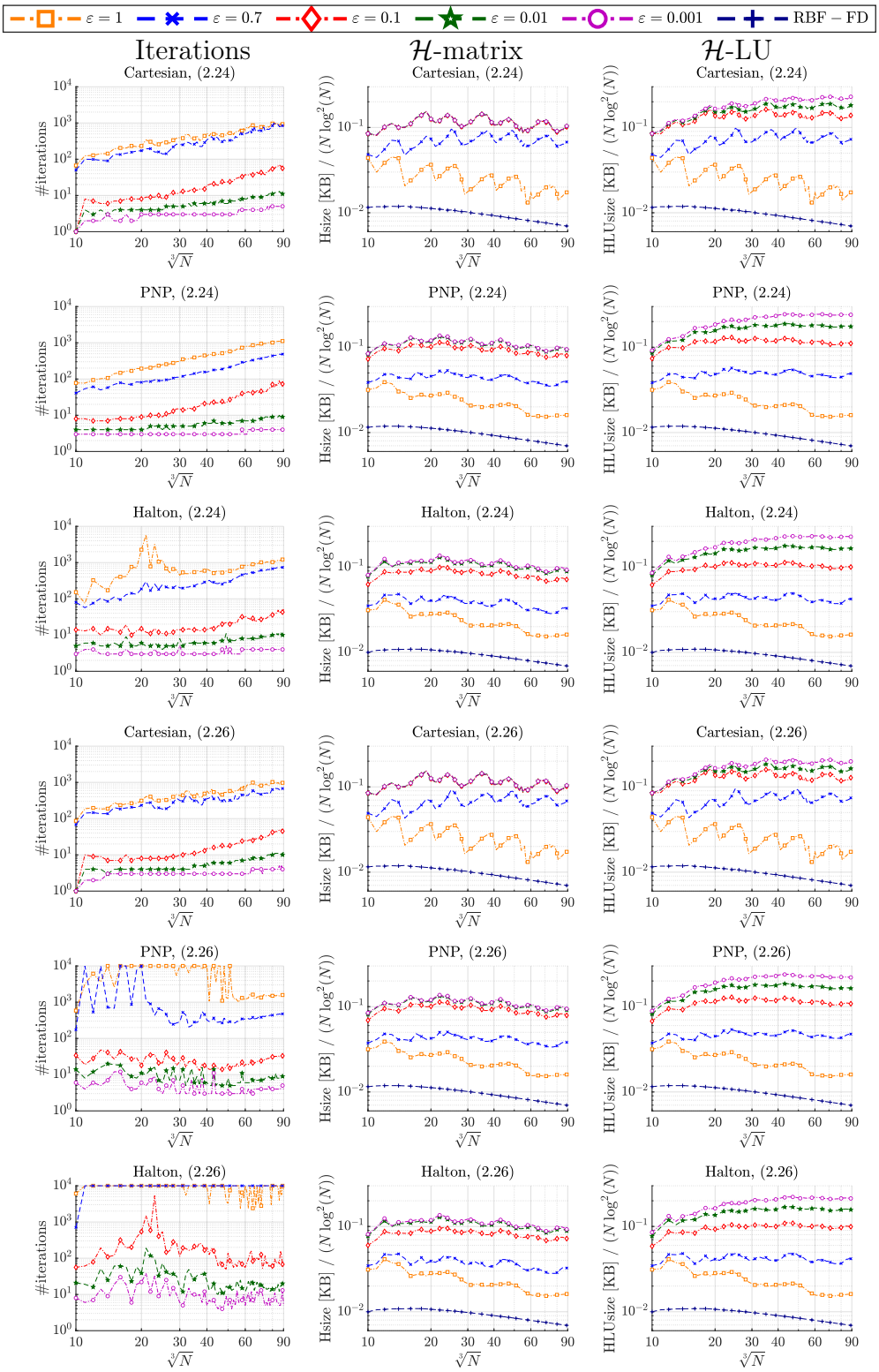

Figure 3.9: Numbers of iterations (left) and scaled memory cost (in KB) for the $\mathcal{H}$-matrix (middle) and the $\mathcal{H}$-LU decomposition (right) for PHS($k = 2,\ \ell = 2$) (2.24) and PHS($k = 3,\ \ell = 2$) (2.26) and recommended preconditioner setup with $n_{\min} = 35$ and tolerances $\varepsilon \in \{1, 0.7, 0.1, 0.01, 0.001\}$ for several node distributions (Cartesian, PNP, Halton).

of the $\mathcal{H}$-matrices and their $\mathcal{H}$-LU factors are for tolerances $\varepsilon < 1$ higher than for setups without coarsening (i. e., above $\mathcal{O}(N \log(N))$, see Remark 3.18). Furthermore, the scaled computation times often stagnate (or increase significantly slower) at $N = 20^3$ for tolerances $\varepsilon < 1$. A comparison (not included here) with the L3 cache size of our hardware revealed that this occurs since the $\mathcal{H}$-LU factors for tolerances $\varepsilon < 1$ fit into the L3 cache up to approximately $N = 20^3$.

Figure 3.10 illustrates the approximation errors (2.27) for the same setups as in Figures 3.8 and 3.9 and in addition for a plain BiCGstab solver without preconditioner as a function of the (unscaled) computation time (in seconds, without the setup time of the nodes and the linear system, i. e., the time for the $\mathcal{H}$-LU approaches is given by the accumulated times shown in Figure 3.8). This demonstrates that for PHS($k = 2,\ \ell = 2$) (2.24) (which is also shown in Figure 2.22) all preconditioner setups lead to reliable results, whereas the plain BiCGstab solver on Halton nodes leads three times to NaN ("Not a Number") as approximation error as well as two times to errors above 1 (which is even obtained by the zero vector). Furthermore, the difference in the accumulated computation time to setup the preconditioner and to solve the linear system can be larger by more than a factor of 10 between different solver setups (i. e., the choice whether a preconditioner is utilized as well as the choice of the preconditioner can drastically influence the overall computation time). However, the setup cost of an $\mathcal{H}$-LU preconditioner with a too small tolerance ε is not amortized and $\varepsilon \geq 0.1$ should hold if the linear system is solved for solely one right hand side. The computation time is clearly the lowest for the plain BiCGstab solver for Cartesian and PNP nodes, whereas an $\mathcal{H}$-LU preconditioner with tolerance $\varepsilon = 1$ or a slightly smaller tolerance can be superior for Halton nodes. The results for PHS($k = 3,\ \ell = 2$) (2.26) (which is also shown in Figures 2.20 and 2.21) are almost identical on Cartesian nodes, except that for smaller errors the gap between the plain BiCGstab solver with tolerance $\varepsilon = 1$ and the other side with tolerances $\varepsilon < 1$, is slightly smaller. However, for PNP and Halton nodes, there are on one hand, (more) "outliers" (i. e., errors corresponding to a not converged solution since a comparison between Figures 2.21 and 2.22 reveals that the condition numbers can be higher for (2.26)). Furthermore, there are more NaN values for the plain BiCGstab solver on Halton nodes and in addition NaN values for $\varepsilon = 1$ on Halton and PNP nodes as well as for the plain BiCGstab solver on PNP nodes. Moreover, the tolerance $\varepsilon = 1$ leads on Halton nodes always to errors above 1 (i. e., no marker is visible). Hence, the fastest computations on PNP and Halton nodes are frequently attained by the tolerance $\varepsilon = 0.1$ and occasionally even by a tolerance $\varepsilon \in \{0.01, 0.001\}$. On the other hand, the larger PHS degree can significantly reduce the errors (i. e., lower errors are achievable, see Figures 2.20 and 2.22). Hence, (2.26) seems to be the better option if a low error should be obtained whereas (2.24) seems to be superior if a fast solution (with typically a higher error) is sufficient.

We observed in the previous tests (Section 3.3 and this subsection) that $\mathcal{H}$-LU preconditioners are especially beneficial for larger problem sizes N if larger PHS degrees are used (2.26), if the linear system should be solved for several right hand sides (i. e., the solution time would be scaled by the number of right hand sides), or if more irregu-

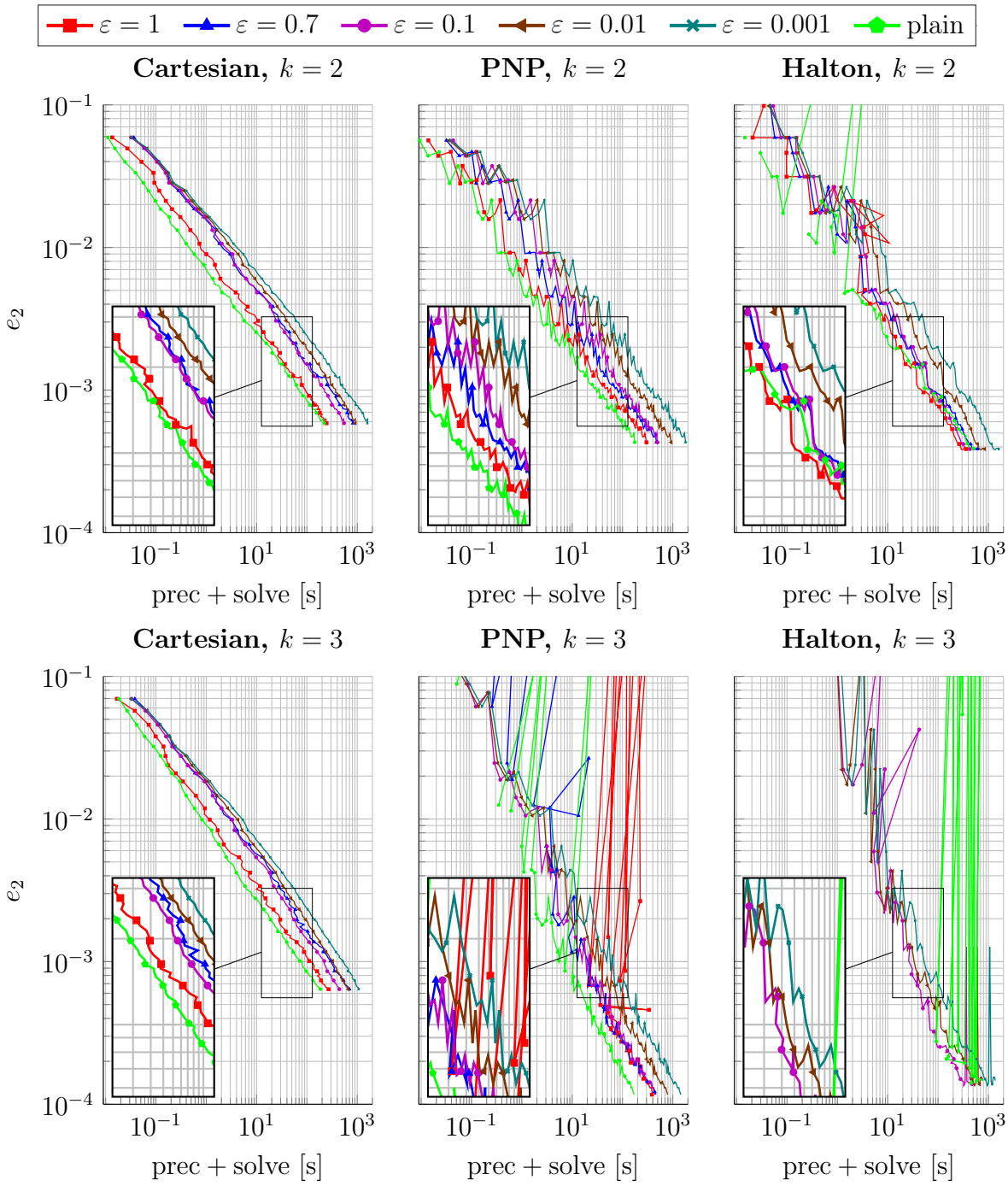

Figure 3.10: Approximation errors (2.27) as a function of the computation time (in seconds, without setup time of the nodes and the linear system) for PHS($k = 2$, $\ell = 2$) (2.24) (top) and PHS($k = 3$, $\ell = 2$) (2.26) (bottom) and recommended preconditioner setup with $n_{\min} = 35$ and tolerances $\varepsilon \in \{1, 0.7, 0.1, 0.01, 0.001\}$ as well as the plain BiCGstab solver on Cartesian (left), PNP (middle) and Halton nodes (right).

lar node distributions are utilized since the influence of the node distribution on the computation time is negligible for sufficiently small ε. Therefore, now we focus on PNP nodes as well as mainly on smaller PHS degrees (2.24) (which will be partly compared to (2.26)) and discuss modifications of our recommended $\mathcal{H}$-LU preconditioner setup. The different tested setups are shown in Table 3.3 in which setup (a) is our recommended setup (see Remark 3.20). The setup (b) uses a smaller $n_{\min} = 20$, whereas (c) uses a larger $n_{\min} = 50$. The setup (d) computes the ranks in the truncation via (3.6), i. e., the relative errors in the Frobenius norm (and results with absolute errors are not shown since they hinder the determination of the tolerance ε and are otherwise comparable). Accumulated updates [14] (lrdecomp2_hmatrix in the H2Lib [52]) are a different approach to compute the $\mathcal{H}$-LU decomposition and are utilized in the setup (e). The setup (f) is without coarsening. The conversion of zero full matrix blocks to rank-0 matrices is avoided in (g). The combination of (f) and (g) is (h). The setups (i) and (j) are similar to (h) except that the weak admissibility condition is replaced by the η-admissibility condition (with $\eta = 15$ in (i) and $\eta = 3$ in (j)).

name	differences to setup (a) with $n_{\min} = 35$: (see Remark 3.20)
(b)	$n_{\min} = 20$
(c)	$n_{\min} = 50$
(d)	with Frobenius norm
(e)	with accumulated updates
(f)	no coarsening
(g)	no conversion of zero full blocks
(h)	no coarsening, no conversion of zero full blocks
(i)	15-admissibility condition, no coarsening, no conversion of zero full blocks
(j)	3-admissibility condition, no coarsening, no conversion of zero full blocks

Table 3.3: Differences between the recommended $\mathcal{H}$-LU setup (a) for polynomial degree $\ell = 2$ and the other tested setups.

Figure 3.11 shows the influence of $n_{\min}$, the norm and the accumulated updates (i. e., the first four modifications in Table 3.3) on the scaled computation times for fixed tolerances $\varepsilon \in \{1, 0.1\}$ and PHS($k = 2,\ \ell = 2$) (2.24). Each color/marker represents a different preconditioner setup and each row corresponds to a different tolerance ε. The results for setup (a) are also included in the second row of Figure 3.8. The left column shows the computation times of the $\mathcal{H}$-matrix. This illustrates that the setups (a), (d) and (e) lead to (almost) identical computation times of the $\mathcal{H}$-matrix (since the corresponding computations are identical in the top plot, whereas only (a) and (e) lead to identical computations in the bottom plot because the rank distribution can be influenced by the norm for $\varepsilon < 1$). Hence, this confirms the statement in Section 3.2 that the differences between different runs of the same computation are typically relatively small compared to the differences between the compared setups. Furthermore, this shows that the choice of the norm for coarsening is negligible. Additionally, it can be observed that $n_{\min} = 35$ leads to the highest computation times for the tolerance $\varepsilon = 1$, i. e., the smaller $n_{\min} = 20$

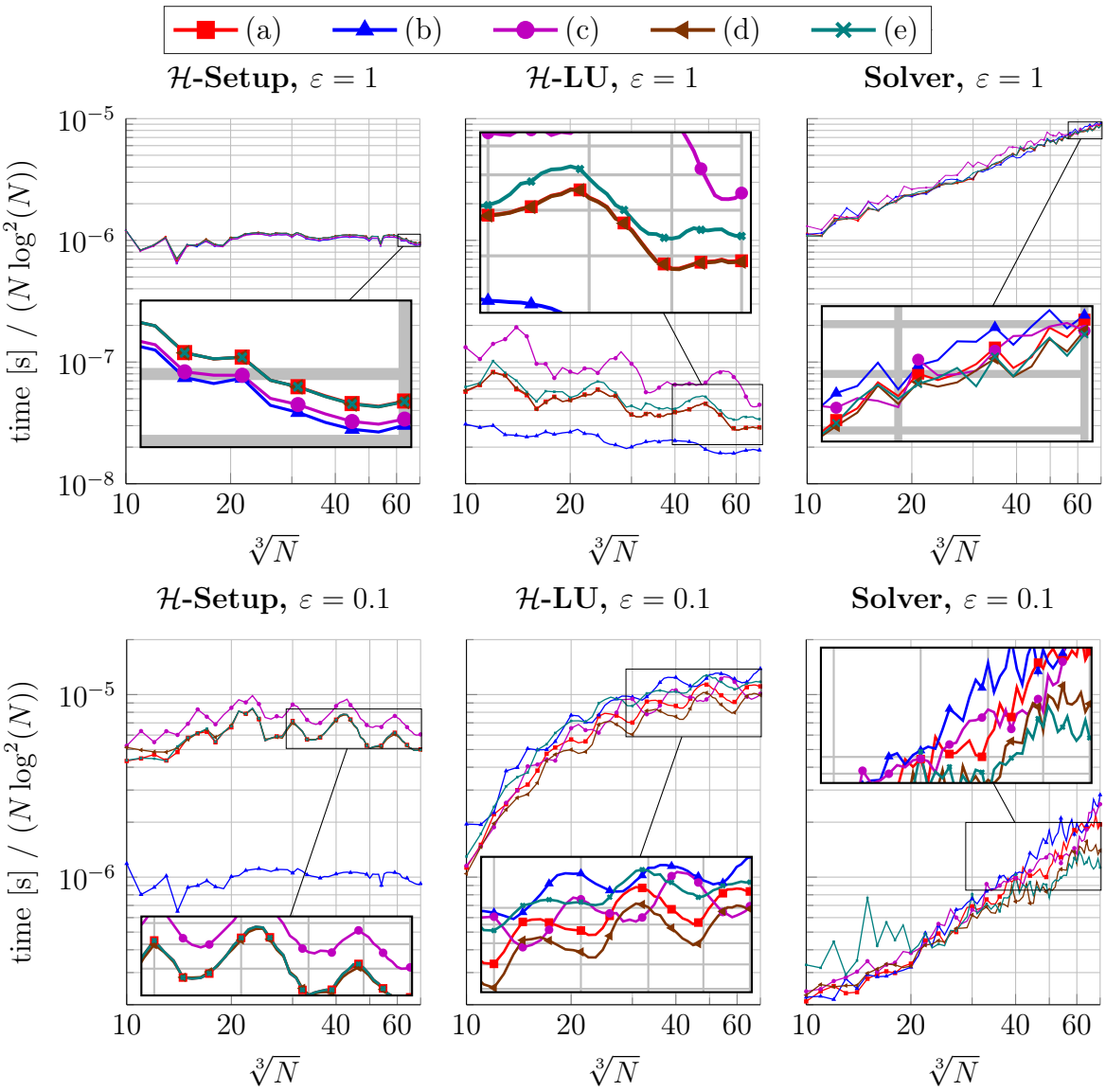

Figure 3.11: Scaled computation times (in seconds) to setup the $\mathcal{H}$-matrix (left) and the $\mathcal{H}$-LU decomposition (middle) and to solve the linear system (right) for PHS($k = 2,\ \ell = 2$) (2.24), PNP nodes, preconditioner setups (a)-(e) (see Table 3.3) with tolerances $\varepsilon \in \{1, 0.1\}$.

as well as the larger $n_{\min} = 50$ lead to slightly lower computation times. However, the plot for the tolerance $\varepsilon = 0.1$ indicates that the larger $n_{\min}$, the larger is the setup time of the $\mathcal{H}$-matrix since a larger $n_{\min}$ leads to larger full matrix blocks (i. e., to more zero entries in the full matrix blocks and to larger matrix blocks during coarsening).

The middle column shows the computation times of the $\mathcal{H}$-LU decomposition. These plots (as well as further tests not included here) indicate that accumulated updates (i. e., setup (e)) typically lead to similar or even higher computation times than the standard

$\mathcal{H}$-LU decomposition. The plot for the tolerance $\varepsilon = 1$ shows that the setups (a) and (d) lead to (almost) identical results since the computations are identical because all ranks are zero (3.15). Furthermore, it can be observed that the larger $n_{\min}$, the larger is the computation time of the $\mathcal{H}$-LU decomposition since a larger $n_{\min}$ leads to larger full matrix blocks. However, the bottom plot for the tolerance $\varepsilon = 0.1$ demonstrates that the setup (b) with the smallest $n_{\min}$ often leads to the highest computation times, whereas the Frobenius norm in setup (d) often leads to the lowest computation times.

The right column shows the solution times of the linear system. This indicates, on one hand, that the setups (b) and (c) often lead to the highest solution times (i. e., a too small as well as a too large $n_{\min}$ can increase the solution time). On the other hand, there is no clear winner (i. e., several setups lead frequently to the lowest solution time). These results confirm that the influence of $n_{\min}$, the norm and accumulated updates on the total computation time is typically relatively small. Furthermore, a reduction in one of these computations (e. g., in the setup time of the $\mathcal{H}$-matrix) often leads to an increase in another computation (e. g., in the $\mathcal{H}$-LU decomposition or in the solution of the linear system).

Figure 3.12 shows the numbers of iterations and the scaled memory cost of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU factors for the different preconditioner setups (a)-(e). The left column shows the numbers of iterations and reveals that the setup (b) usually leads (for both tolerances ε) to higher numbers of iterations than the setup (a), whereas the setup (a) often leads to higher numbers of iterations than the setup (c). This indicates that a decrease in $n_{\min}$ usually leads to an increase in the numbers of iterations, whereas an increase in $n_{\min}$ often leads to a decrease in the numbers of iterations. The setups (a) and (d) lead to identical numbers of iterations for the tolerance $\varepsilon = 1$ (since the different norms lead to identical $\mathcal{H}$-LU decompositions), whereas the accumulated updates in setup (e) often lead to slightly different numbers of iterations than the setup (a). However, for the tolerance $\varepsilon = 0.1$, the setup (e) leads for smaller N to the largest numbers of iterations and for larger N to the smallest numbers of iterations, whereas the Frobenius norm in setup (d) usually leads to similar (and frequently to slightly smaller) numbers of iterations than the setup (a). The last two columns demonstrate that a decrease in $n_{\min}$ leads to a decrease in the memory cost, whereas an increase in $n_{\min}$ leads to an increase in the memory cost. Furthermore, it can be observed for the tolerance $\varepsilon = 0.1$ that the setup (d) leads to slightly smaller memory cost than the setup (a), whereas the setups (a) and (e) lead to (almost) identical memory cost of the $\mathcal{H}$-LU decomposition. However, these results in Figure 3.12 again illustrate (as in Figure 3.11) that the influence of these modifications is negligible compared to the influence of the tolerance ε (see Figure 3.9).

Figure 3.13 is similar to Figure 3.11 except that the focus is now on the comparison of setup (a) with the setups (f) to (j) (i. e., the last five modifications in Table 3.3). This shows the influence of the admissibility condition (weak or η), of coarsening and of the conversion of all zero full matrix blocks in the $\mathcal{H}$-matrix to rank-0 matrices on the scaled computation times for fixed tolerances $\varepsilon \in \{1, 0.1\}$.

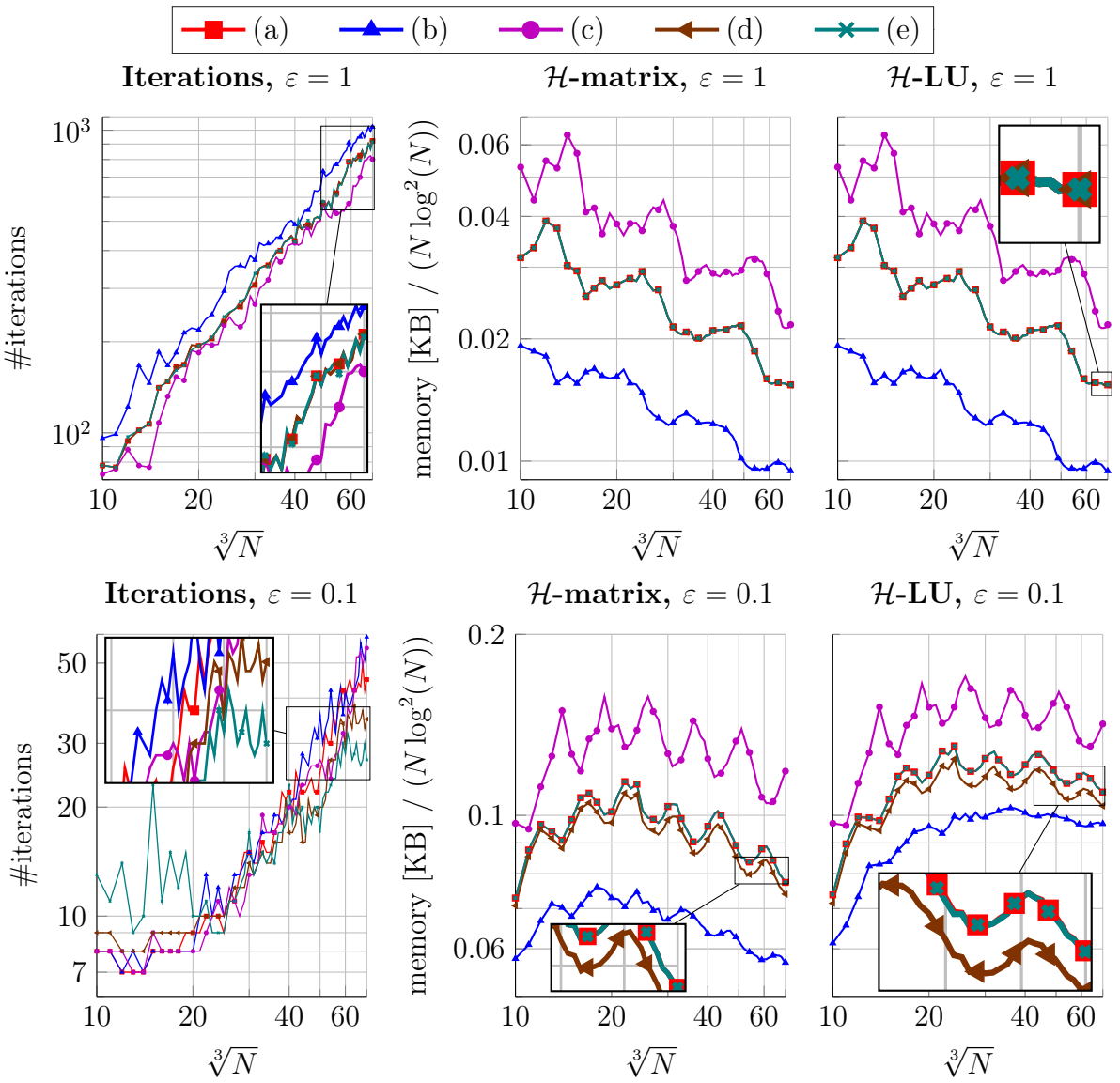

Figure 3.12: Numbers of iterations (left) and scaled memory cost (in KB) for the $\mathcal{H}$-matrix (middle) and the $\mathcal{H}$-LU decomposition (right) for PHS($k = 2$, $\ell = 2$) (2.24), PNP nodes, preconditioner setups (a)-(e) (see Table 3.3) with tolerances $\varepsilon \in \{1, 0.1\}$.

The results for setup (a) are in several plots (almost) identical to the results for setup (g) (and therefore hardly visible in these plots). The reason for these (almost) identical results is that the conversion of zero full matrix blocks to rank-0 matrices is not necessary if the tolerance $\varepsilon = 1$ is utilized (since the obtained $\mathcal{H}$-matrix is not changed). Hence, the comparison between the setups (a) and (g) shows for the tolerance $\varepsilon = 1$ the computational cost of the conversion of zero full matrix blocks to rank-0 matrices (left plot), whereas the middle and right plot illustrate the difference between different runs of an identical computation.

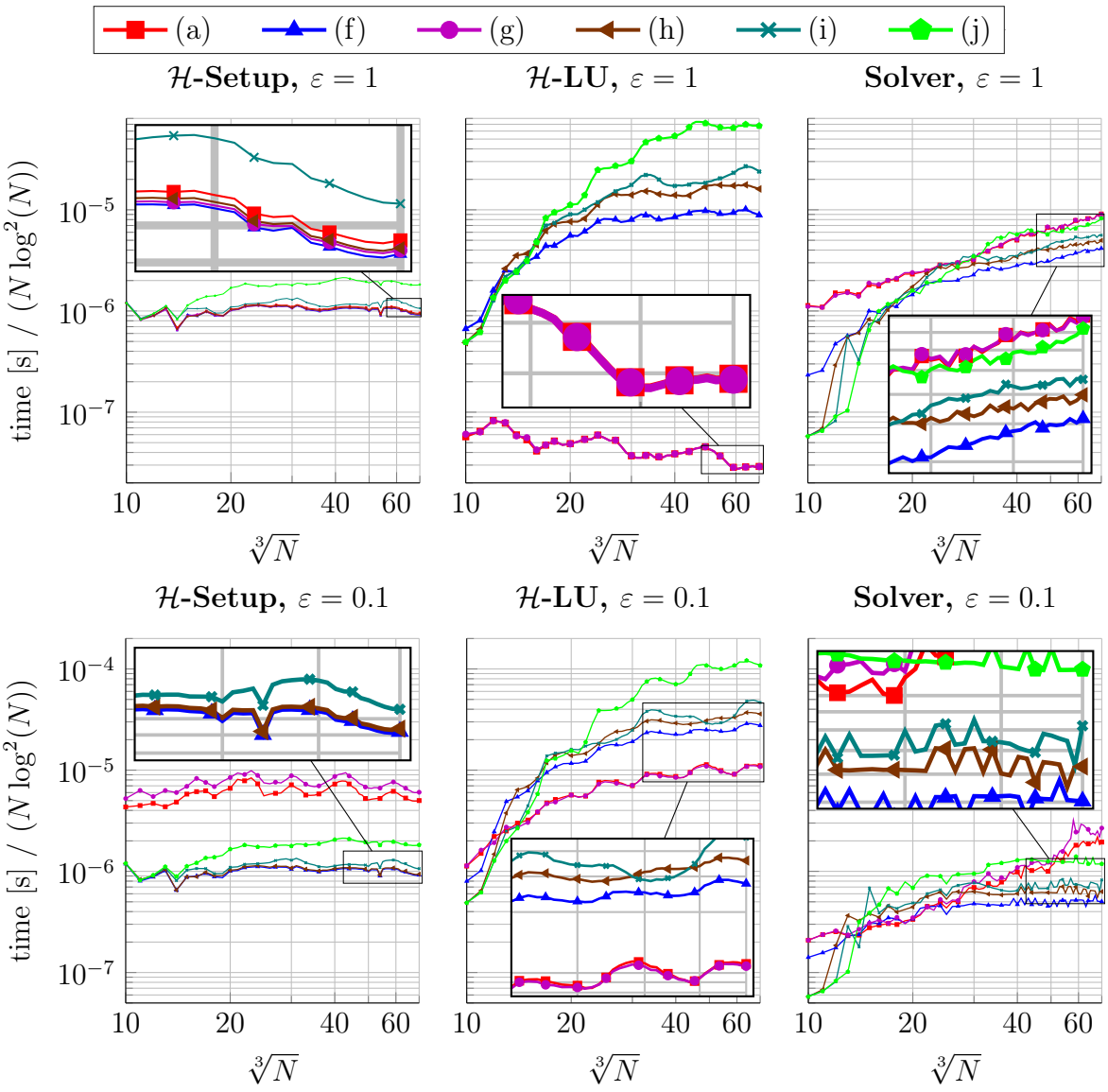

Figure 3.13: Scaled computation times (in seconds) to setup the $\mathcal{H}$-matrix (left) and the $\mathcal{H}$-LU decomposition (middle) and to solve the linear system (right) for PHS($k = 2$, $\ell = 2$) (2.24), PNP nodes, preconditioner setups (a), (f)-(j) (see Table 3.3) with tolerances $\varepsilon \in \{1, 0.1\}$.

The influence of these modifications on the setup time of the $\mathcal{H}$-matrix is shown in the left column. The top plot (for tolerance $\varepsilon = 1$) reveals that the setups (j) and (i) lead to the highest and second highest setup times, resp. This demonstrates that using an η-admissibility condition increases the setup time and the smaller η, the higher is the setup time. However, all other setups (a), (f), (g) and (h) lead to (almost) identical setup times. Furthermore, the bottom plot (for tolerance $\varepsilon = 0.1$) illustrates that coarsening (which is used in the setups (a) and (g)) significantly increases the setup time of the $\mathcal{H}$-matrix (if a tolerance $\varepsilon < 1$ is utilized). This demonstrates that the setup of R_b for

each block b satisfying (3.14) and the computation of its truncation $\mathcal{T}^{\mathcal{R}}(R_b)$ are time consuming. Coarsening leads even to a higher setup time than using an η-admissibility condition. However, a prior conversion of zero full matrix blocks to rank-0 matrices, i. e., using setup (a) instead of (g) can speedup the coarsening and with that reduce the setup time of the $\mathcal{H}$-matrix. This reduction in the setup time is significantly higher than the almost invisible difference between the setups (f) and (h). This illustrates that the conversion of zero full matrix blocks to rank-0 matrices significantly accelerates coarsening, whereas its additional computational cost is negligible.

The middle column indicates that the computation time of the $\mathcal{H}$-LU decomposition can vary significantly between the tested setups. This computation time is usually (i. e., in all cases with the exception of the combination of the tolerance $\varepsilon = 0.1$ with a small N) significantly reduced if coarsening is performed (i. e., setup (a) or (g)). This reduction in the computation time of the $\mathcal{H}$-LU decomposition decreases for decreasing tolerance ε. The comparison between (f) and (h) demonstrates that the conversion of zero full matrix blocks to rank-0 matrices can speedup the $\mathcal{H}$-LU decomposition (if no coarsening was done and a sufficiently large N in combination with a not too small tolerance ε are utilized). Moreover, this indicates that zero full matrix blocks can occur when using the weak admissibility condition. Furthermore, it can be observed that (for a sufficiently large N value) the setup (i) (i. e., the η-admissibility condition with a larger η value) frequently leads to higher computation times than the setup (h), whereas setup (j) (i. e., the η-admissibility condition with a smaller η value) significantly increases the setup time of the $\mathcal{H}$-LU decomposition.

The results in the right column show the influence of these modifications on the solution time and indicate that the variants without coarsening often lead to the highest solution time. Furthermore, the setup (f) (i. e., with conversion of zero full matrix blocks to rank-0 matrices but without coarsening) leads for sufficiently large N usually to the smallest solution time. This illustrates that the "false" nonzero entries (see Figure 3.6) can lead to an increase in the solution time. All in all, it can be observed that the η-admissibility condition in particular, and also other setups without coarsening, lead to significantly increased setup times of the preconditioner without a significant reduction in the solution time. This additionally demonstrates that the cost of coarsening and of the conversion of zero full matrix blocks to rank-0 matrices is well invested as long as the tolerance ε is not too small.

Figure 3.14 is similar to Figure 3.12 and shows the numbers of iterations and the scaled memory cost of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU factors for the setups used in Figure 3.13. For the tolerance $\varepsilon = 1$ (top row), the results for setup (a) are identical to the results for setup (g) (and therefore hardly visible). This confirms that the conversion of zero full matrix blocks to rank-0 matrices is not necessary if the tolerance $\varepsilon = 1$ is utilized since all zero full matrix blocks satisfy (3.14). Furthermore, it can be observed that coarsening (in particular, with the tolerance $\varepsilon = 1$) leads to a significant increase in the numbers of iterations as well as a significant decrease in the memory cost. Moreover, it can be seen that there is a clear separation between the different setups for the tolerance

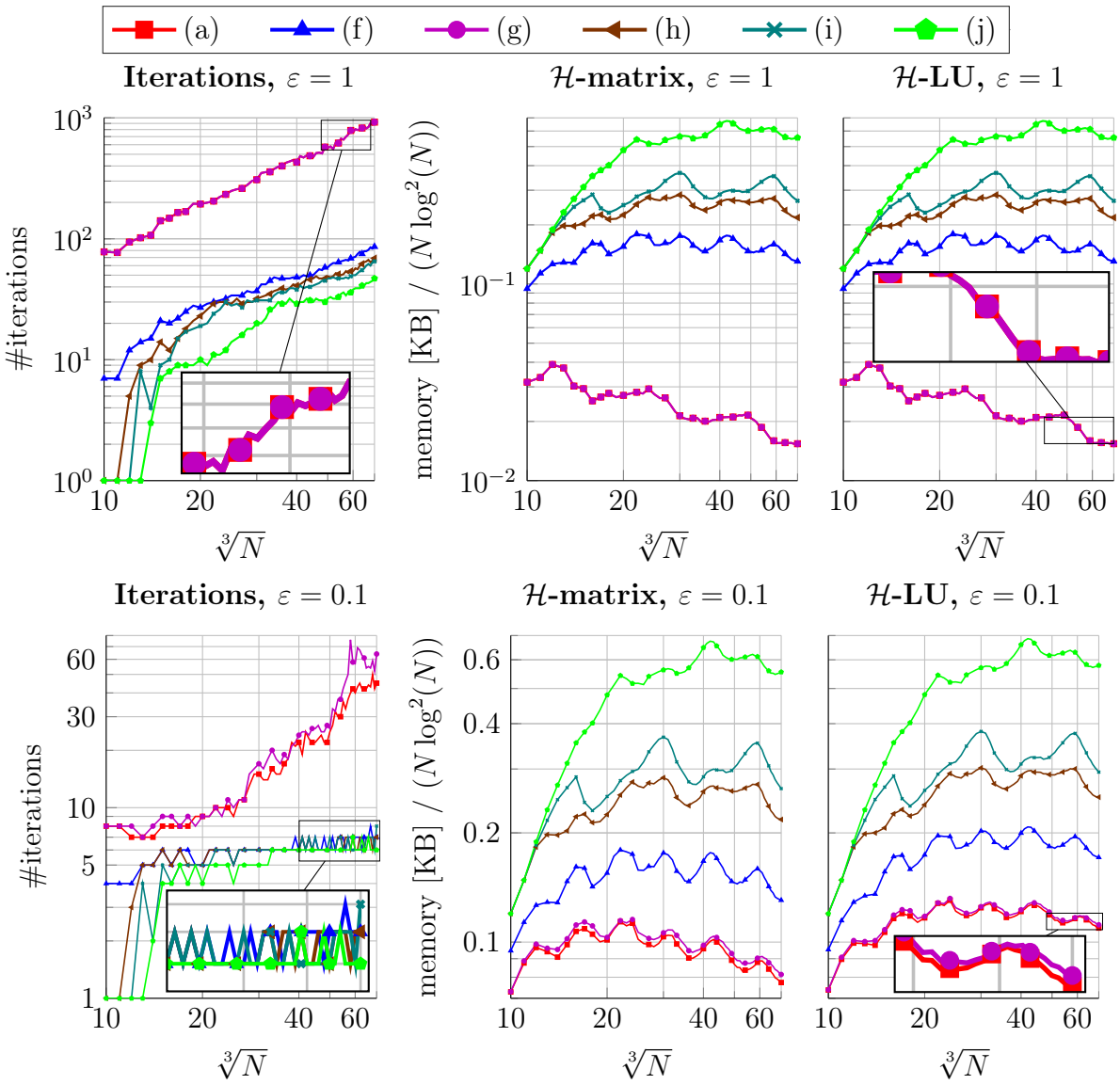

Figure 3.14: Numbers of iterations (left) and scaled memory cost (in KB) for the $\mathcal{H}$-matrix (middle) and the $\mathcal{H}$-LU decomposition (right) for PHS($k = 2$, $\ell = 2$) (2.24), PNP nodes, preconditioner setups (a), (f)-(j) (see Table 3.3) with tolerances $\varepsilon \in \{1, 0.1\}$.

$\varepsilon = 1$ (with the exception that (a) and (g) lead to identical results) and the higher the numbers of iterations, the lower are the corresponding memory cost. Furthermore, each of the following decisions leads to a reduction in the memory cost (i. e., to an increase in the numbers of iterations)

- increasing the η value in the η-admissibility condition,
- using a weak instead of an η-admissibility condition,

- converting zero full matrix blocks to rank-0 matrices,
- using coarsening.

However, the plots for the smaller tolerance $\varepsilon = 0.1$ (bottom row) demonstrate that only coarsening leads to a significant increase in the numbers of iterations, whereas the other setups lead to almost identical numbers of iterations. Nevertheless, the plots for the memory cost of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU decomposition indicate that the significant differences (observed for the tolerance $\varepsilon = 1$) between the various setups are still present if a smaller tolerance ε is utilized. This illustrates that, on one hand, a weak admissibility condition with conversion of zero full matrix blocks to rank-0 matrices should be used (as long as the tolerance ε is not significantly smaller than 0.1). Furthermore, this explains the observations in Figure 3.13 that the setup (f) leads for sufficiently large N usually to the lowest solution times (since the lower the memory cost of the $\mathcal{H}$-LU decomposition, the lower is the computational cost per iteration). However, on the other hand, the effect of the conversion of zero full matrix blocks to rank-0 matrices depends significantly on the tolerance ε used.

3.4.6 Numerical results for larger degrees of polynomial augmentation

The results of this subsection are presented through the following figures to illustrate whether the results from the previous Subsection 3.4.5 generalize to higher polynomial degrees $\ell \geq 5$.

- Figure 3.15: Influence of the problem size N on the computation time for different preconditioner setups with truncation tolerances $\varepsilon \in \{1, 0.1\}$ for PHS($k = 3,\ \ell = 5$) (2.24).
- Figure 3.16: Influence of the problem size N on the numbers of iterations and the memory cost of different preconditioner setups with truncation tolerances $\varepsilon \in \{1, 0.1\}$ for PHS($k = 3,\ \ell = 5$) (2.24).
- Figure 3.17: Relationship between the computation time and the approximation errors for truncation tolerances $\varepsilon \in \{1, 0.1, 0.001\}$ and PHS($k = 5,\ \ell = 5$) (2.26).
- Figure 3.18: Computation times, memory cost, numbers of iterations and approximation errors for PHS($k = 3,\ \ell = 5$) and PHS($k = 5,\ \ell = 5$) and truncation tolerances $\varepsilon \in \{1, 0.1, 0.01\}$.
- Figure 3.19: Influence of the problem size N on the computation time for truncation tolerances $\varepsilon \in \{1, 0.1\}$, several $n_{\min}$ and PHS($k = 4,\ \ell = 8$) (2.24).
- Figure 3.20: Influence of the problem size N on the numbers of iterations and the memory cost of preconditioners with truncation tolerances $\varepsilon \in \{1, 0.1\}$, several $n_{\min}$ and PHS($k = 4,\ \ell = 8$) (2.24).
- Figure 3.21: Relationship between the computation time and the approximation

errors for truncation tolerances $\varepsilon \in \{1, 0.1, 0.001\}$, several $n_{\min}$ and PHS($k = 4$, $\ell = 8$) (2.24).

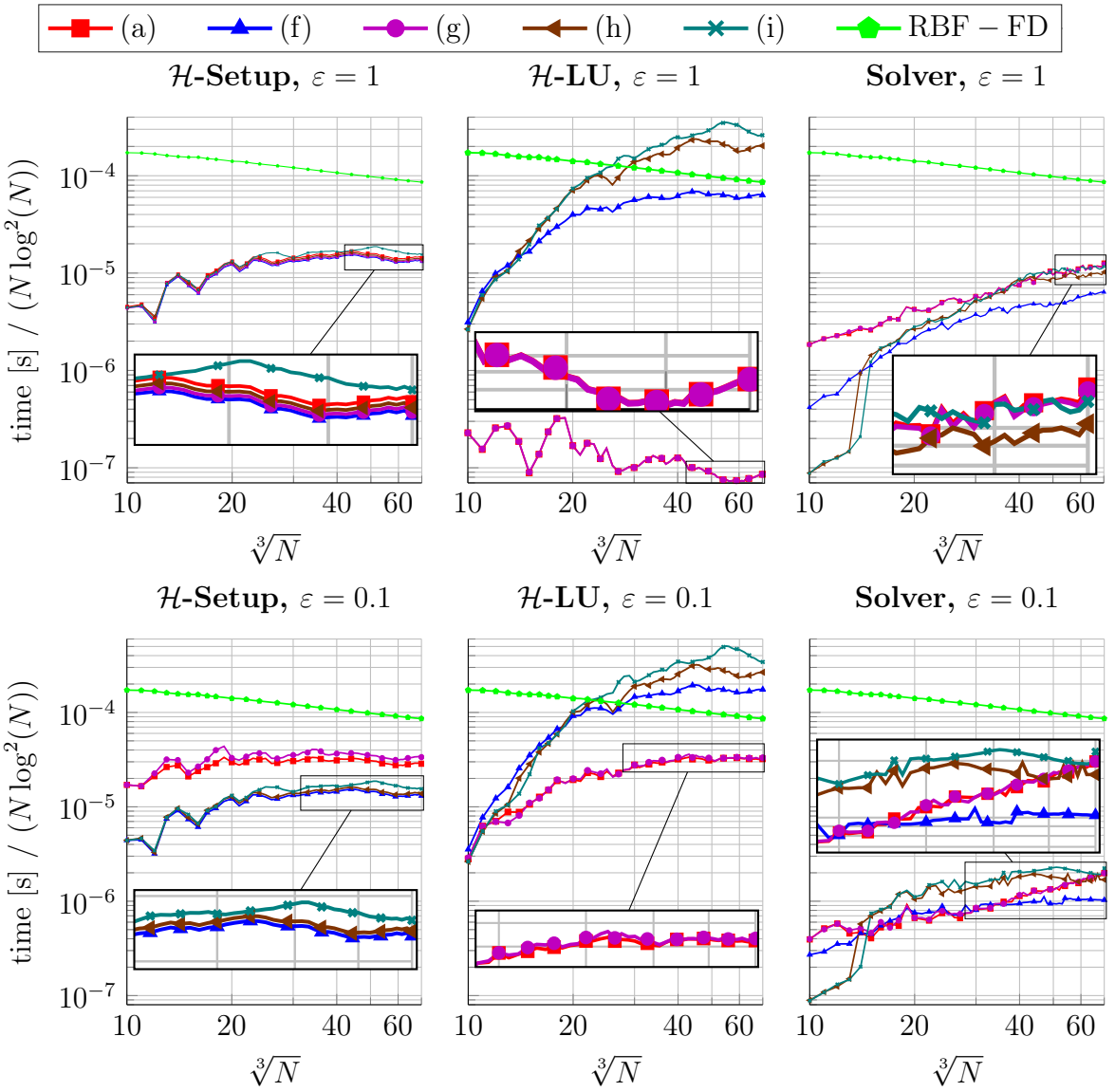

Figure 3.15: Scaled computation times (in seconds) to setup the $\mathcal{H}$-matrix (left) and the $\mathcal{H}$-LU decomposition (middle) and to solve the linear system (right) for PHS($k = 3$, $\ell = 5$) (2.24), PNP nodes, preconditioner setups (a), (f)-(i) with $n_{\min} = 50$ (see Table 3.3) with tolerances $\varepsilon \in \{1, 0.1\}$.

Figure 3.15 is similar to Figure 3.13, except that PHS($k = 3$, $\ell = 5$) (2.24) with $n_{\min} = 50$ is used instead of PHS($k = 2$, $\ell = 2$) with $n_{\min} = 35$ and that no results for the η-admissibility condition with $\eta = 3$ are included. However, we included instead the additional color/marker "RBF-FD" which is the same in all six plots and represents the setup time of the linear system (without the setup time of the node distribution). This curve "RBF-FD" is also shown for PHS($k = 3$, $\ell = 5$) (2.24) with a different scaling factor

in Figure 3.3. The results for setup (a) are again in several plots (almost) identical to the results for setup (g) (and therefore hardly visible in these plots). The results shown in Figures 3.13 and 3.14 demonstrate that the η-admissibility condition significantly increases the computation and memory cost without a significant improvement in the accuracy of the preconditioner. Further tests (not included here) indicate that this increase in the computation and memory cost drastically increases for larger degrees (since larger stencil sizes lead to larger associated sets (3.13) and bounding boxes as well as a higher likelihood of empty sons during the cluster tree computation (see Subsection 3.4.4), i. e., to more and larger full matrix blocks). Therefore, we recommend to utilize the weak admissibility condition (possibly enhanced with coarsening or conversion of zero full matrix blocks to rank-0 matrices) and include for comparison in our tests only the η-admissibility condition with $\eta = 15$.

This Figure 3.15 demonstrates that the results for $\ell = 2$ (i. e., notably the setups with coarsening lead to the overall lowest computation times) also hold for the RBF-FD discretization with $\ell = 5$, whereas the main difference is the increased computation time for the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU decomposition. The comparison between these figures reveals in addition that for the larger degree $\ell = 5$, the difference between the setups with coarsening ((a) and (g)) and without coarsening ((f) and (h)) decreases in terms of the setup time of the $\mathcal{H}$-matrix for the case with tolerance $\varepsilon = 0.1$, although it increases in terms of the setup time of the $\mathcal{H}$-LU decomposition. Reasons for this observation could be that we utilize the symmetrized sparse matrix $B + B^T$ for the sparsity structure as well as we stop the partition of the cluster tree if the case with an empty son occurs in the function build_adaptive_dd_cluster (see Subsection 3.4.4) and the likelihood of this case is significantly higher for our setup with degree $\ell = 5$ (since the stencil size is relatively large compared to the $n_{\min}$ value used). For example, this case does not occur for $\ell = 2$ and $n_{\min} = 35$, whereas $n_{\min} > 200$ would be necessary to avoid this case for the setup with degree $\ell = 5$. A drawback of this approach is therefore that large full matrix blocks can be created even if they consist of only one (falsely assigned, i. e., a nonzero entry of $B + B^T$ which is a zero entry in B) nonzero entry and could be efficiently represented by low-rank approximations. Additionally, this facilitates more unbalanced (block) cluster trees since some clusters are not further partitioned.

Figure 3.16 is similar to Figure 3.14 (except that the color/marker (j) is replaced by the color/marker "RBF-FD" which is only used in the middle and right column and represents the memory cost of the stiffness matrix) and shows the numbers of iterations and the scaled memory cost of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU factors for the setups (a), (f)-(i) with $n_{\min} = 50$. For the tolerance $\varepsilon = 1$ (top row), the results for setup (a) are again identical to the results for setup (g) (and therefore hardly visible). This demonstrates again that the results for $\ell = 2$ generalize to RBF-FD discretizations with larger degrees ℓ, whereas the reduction in memory cost which is gained by coarsening is increased for the larger degree $\ell = 5$ (since more stops due to empty sons occur in the computation of the cluster tree than for $\ell = 2$, see Subsection 3.4.4). Additionally, it can be observed that coarsening in combination with a high tolerance ε (e. g., $\varepsilon = 1$) can even lead to preconditioners with lower memory cost than the stiffness matrix.

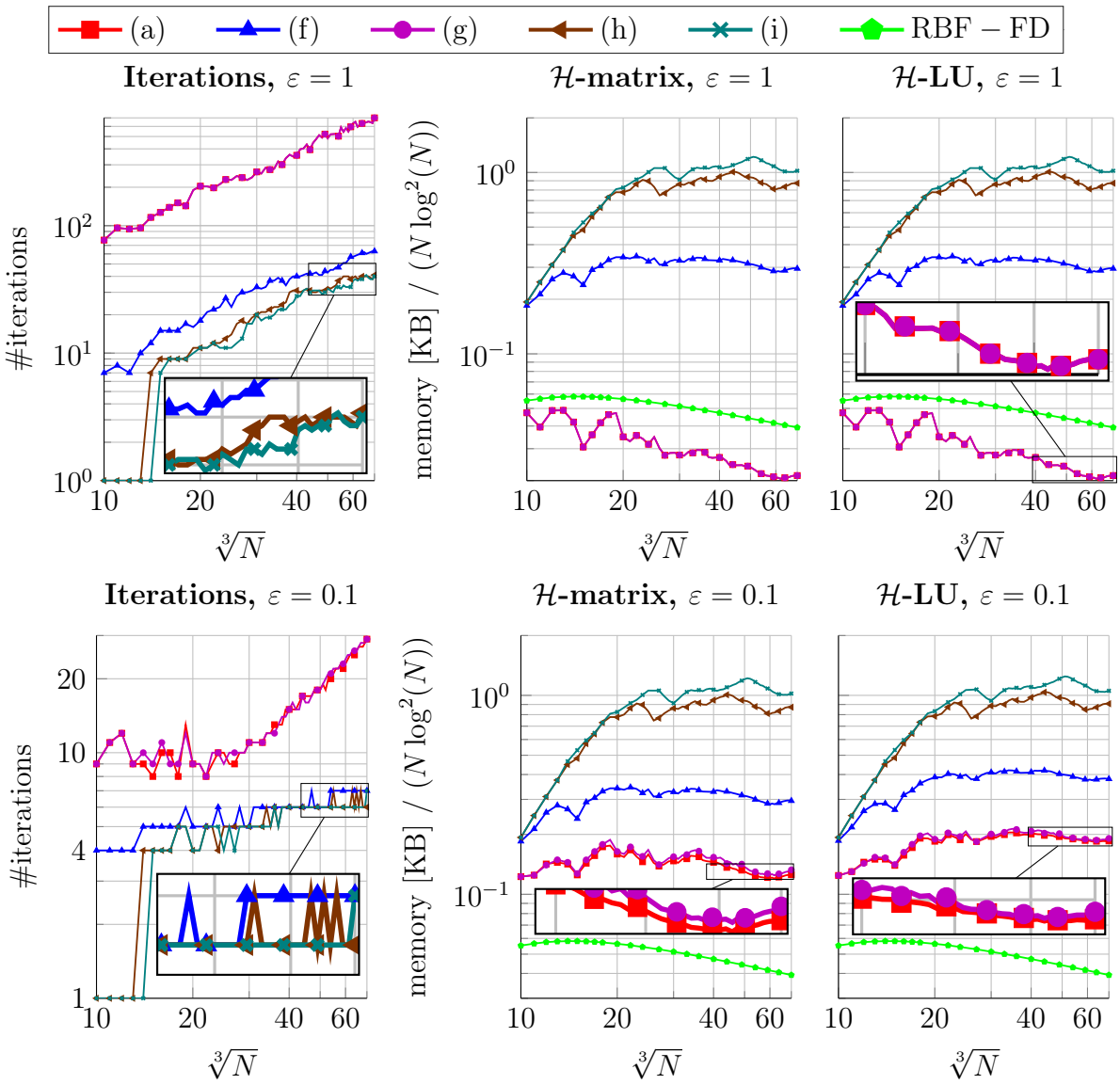

Figure 3.16: Numbers of iterations (left) and scaled memory cost (in KB) for the $\mathcal{H}$-matrix (middle) and the $\mathcal{H}$-LU decomposition (right) for PHS($k = 3$, $\ell = 5$) (2.24), PNP nodes, preconditioner setups (a), (f)-(i) with $n_{\min} = 50$ (see Table 3.3) with tolerances $\varepsilon \in \{1, 0.1\}$.

Figure 3.17 illustrates the approximation errors (2.27) (for the same RBF-FD discretizations and the same preconditioner setups as in Figures 3.15 and 3.16) as a function of the (unscaled) computation time (in seconds, without the setup time of the nodes and the linear system, i. e., the time is given by the accumulated times shown in Figure 3.15). The results for the setup (a) are again frequently (almost) identical to the results for the setup (g) (and therefore hardly visible). Figure 3.17 demonstrates that all preconditioner setups lead to reliable results. However, the difference (between different preconditioner setups) in the accumulated computation time to setup the preconditioner

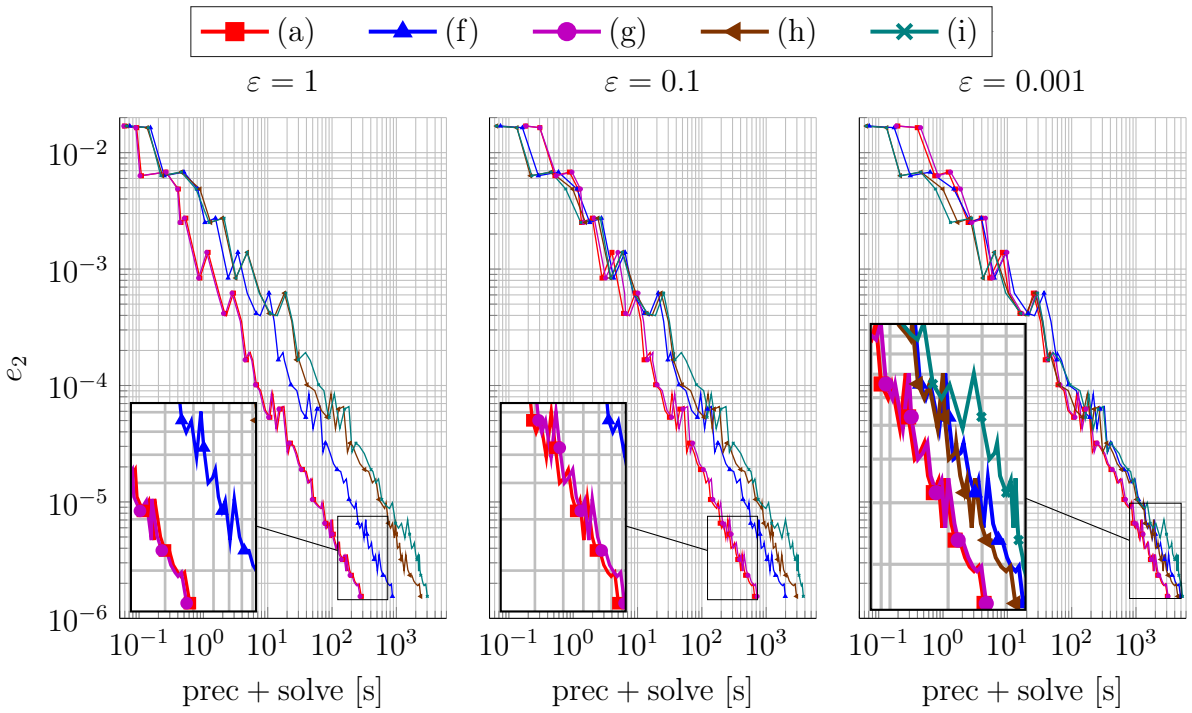

Figure 3.17: Approximation errors (2.27) as a function of the computation time (in seconds, without setup time of the nodes and the linear system) for PHS($k = 3,\ \ell = 5$) (2.24), PNP nodes and preconditioner setups (a), (f)-(i) with $n_{\min} = 50$ (see Table 3.3) with tolerances $\varepsilon \in \{1, 0.1, 0.001\}$.

and to solve the linear system can be larger by more than a factor of 10. The plot in the left column shows that these differences are especially large for the high tolerance $\varepsilon = 1$, whereas they decrease with decreasing tolerance ε. Furthermore, it can be seen that (g) is usually the fastest setup for the tolerance $\varepsilon = 1$ (since the conversion of zero full matrix blocks to rank-0 matrices is not necessary for $\varepsilon = 1$ due to (3.15)), whereas (a) leads often to the fastest results for the smaller tolerances $\varepsilon < 1$ (i. e., the conversion of zero full matrix blocks to rank-0 matrices can reduce the overall computation time). Hence, this indicates that coarsening can still be beneficial if a relatively small tolerance $\varepsilon = 0.001$ is utilized. Nevertheless, the conversion of zero full matrix blocks to rank-0 matrices without coarsening (i. e., using setup (h) instead of (g)) often increases the overall computation time if the smaller tolerance $\varepsilon = 0.001$ is used. However, even the setups (h) and (i) can lead to the fastest results for the smaller tolerances $\varepsilon < 1$ and sufficiently small approximation error (2.27), although these setups typically lead to the highest computation times.

Figure 3.18 shows results for our recommended preconditioner setup with $n_{\min} = 50$ and tolerances $\varepsilon \in \{1, 0.1, 0.01\}$ for PHS($k = 3,\ \ell = 5$) and PHS($k = 5,\ \ell = 5$). Each color/marker represents a different combination of k and ε values. The results for PHS($k = 3,\ \ell = 5$) with $\varepsilon \in \{1, 0.1\}$ are also shown in Figures 3.15, 3.16 and 3.17 as setup (a).

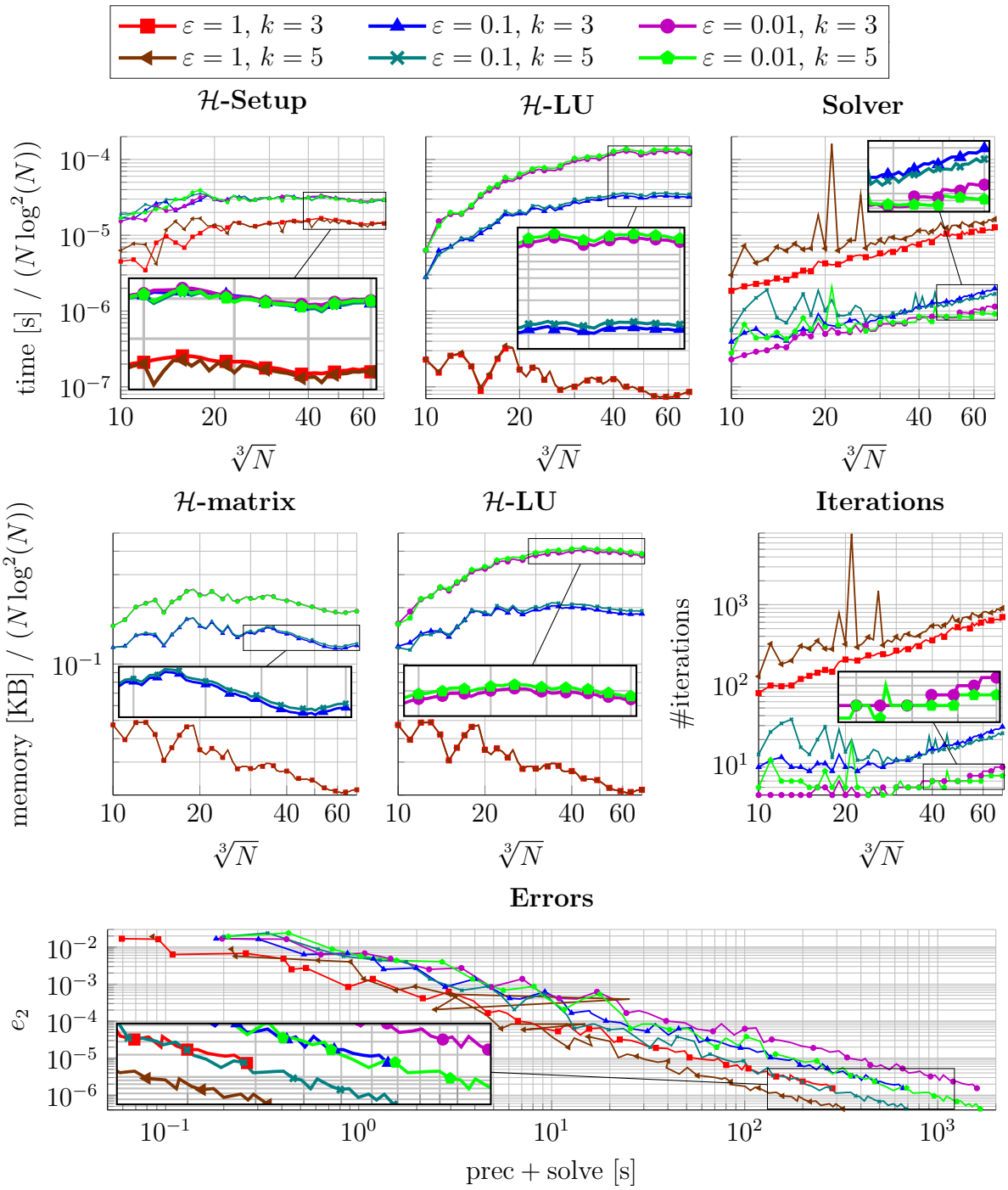

Figure 3.18: Scaled computation times (in seconds, top), scaled memory cost (in KB) and numbers of iterations (middle), and approximation errors (2.27) (bottom) for PHS($k = 3$, $\ell = 5$) (2.24) and PHS($k = 5$, $\ell = 5$) (2.26) on PNP nodes and recommended preconditioner setup with $n_{\min} = 50$ and tolerances $\varepsilon \in \{1, 0.1, 0.01\}$.

The first row is similar to Figure 3.15 and contains scaled computation times. The left plot illustrates that the setup time of the $\mathcal{H}$-matrix is mainly influenced by the tolerance ε, and the setup time oscillates w. r. t. the degree k for smaller N values. The middle plot shows the computation times of the $\mathcal{H}$-LU decomposition and reveals that a higher k value can slightly increase the computation time of the $\mathcal{H}$-LU decomposition. However, the influence of k is relatively small compared to the influence of the tolerance ε. Furthermore, the computation times of the setups with $\varepsilon = 1$ are (almost) identical since the memory cost of the corresponding $\mathcal{H}$-matrices and their $\mathcal{H}$-LU factors does not depend on k (see Remark 3.13). The right plot contains the solution times of the linear system and demonstrates that the solution time is mainly influenced by the tolerance ε. Nevertheless, the (oscillations in the) solution time typically increases with the k value (in particular for smaller N values).

The second row is similar to Figure 3.16 and contains scaled memory cost and numbers of iterations. The left and middle plots indicate that the memory cost of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU factors depends mainly on the tolerance ε. As mentioned above, k has no influence on the memory cost of the tolerance $\varepsilon = 1$. However, a larger k value leads often to a slightly higher memory cost of the tolerances $\varepsilon < 1$. The right plot illustrates that the number of iterations is mainly influenced by the tolerance ε. Nevertheless, the (oscillations in the) number of iterations typically increases with the k value (in particular for smaller N values).

The plot in the bottom row is similar to Figure 3.17 and contains the approximation errors (2.27). A NaN value is only computed once for the setup with $\varepsilon = 1$ and $k = 5$. Hence, the performance of the $\mathcal{H}$-LU preconditioner with tolerance $\varepsilon = 1$ seems to be more stable for the larger polynomial degree $\ell = 5$ than for the smaller polynomial degree $\ell = 2$ (see Figure 3.10). Furthermore, it can be seen that the fastest solution corresponds again to the tolerance $\varepsilon = 1$, i. e., the setup cost of the preconditioners with smaller tolerances $\varepsilon < 1$ is not amortized, and the larger k value (2.26) usually reaches a given approximation error faster than the smaller k value (2.24).

Computation times for our recommended $\mathcal{H}$-LU preconditioner setup with tolerances $\varepsilon \in \{1, 0.1\}$ are shown in Figure 3.19 for PHS($k = 4,\ \ell = 8$) (2.24). Each color/marker corresponds to a different $n_{\min} \in \{40, 70, 100, 200, 300\}$ (and there is in addition an "RBF-FD" color/marker as in Figure 3.15). Figure 3.19 is similar to Figure 3.11 except that the degree ℓ is larger, only the influence of $n_{\min}$ is shown and results are solely computed up to $N = 53^3$. The left column shows the setup times of the $\mathcal{H}$-matrix and confirms our observations in Figure 3.11 for the tolerance $\varepsilon = 0.1$, namely the larger $n_{\min}$, the larger is the setup time of the $\mathcal{H}$-matrix. However, this observation holds for the larger degree $\ell = 8$ also for the tolerance $\varepsilon = 1$. Furthermore, the middle and the right columns confirm that our observations in Figure 3.11 about the computation time of the $\mathcal{H}$-LU decomposition and about the solution time hold as well for the larger degree $\ell = 8$. In particular, the larger $n_{\min}$, the larger is the computation time of the $\mathcal{H}$-LU decomposition (if the tolerance $\varepsilon = 1$ is used), whereas the highest computation time of the $\mathcal{H}$-LU decomposition is usually given by the smallest $n_{\min}$ value (if the

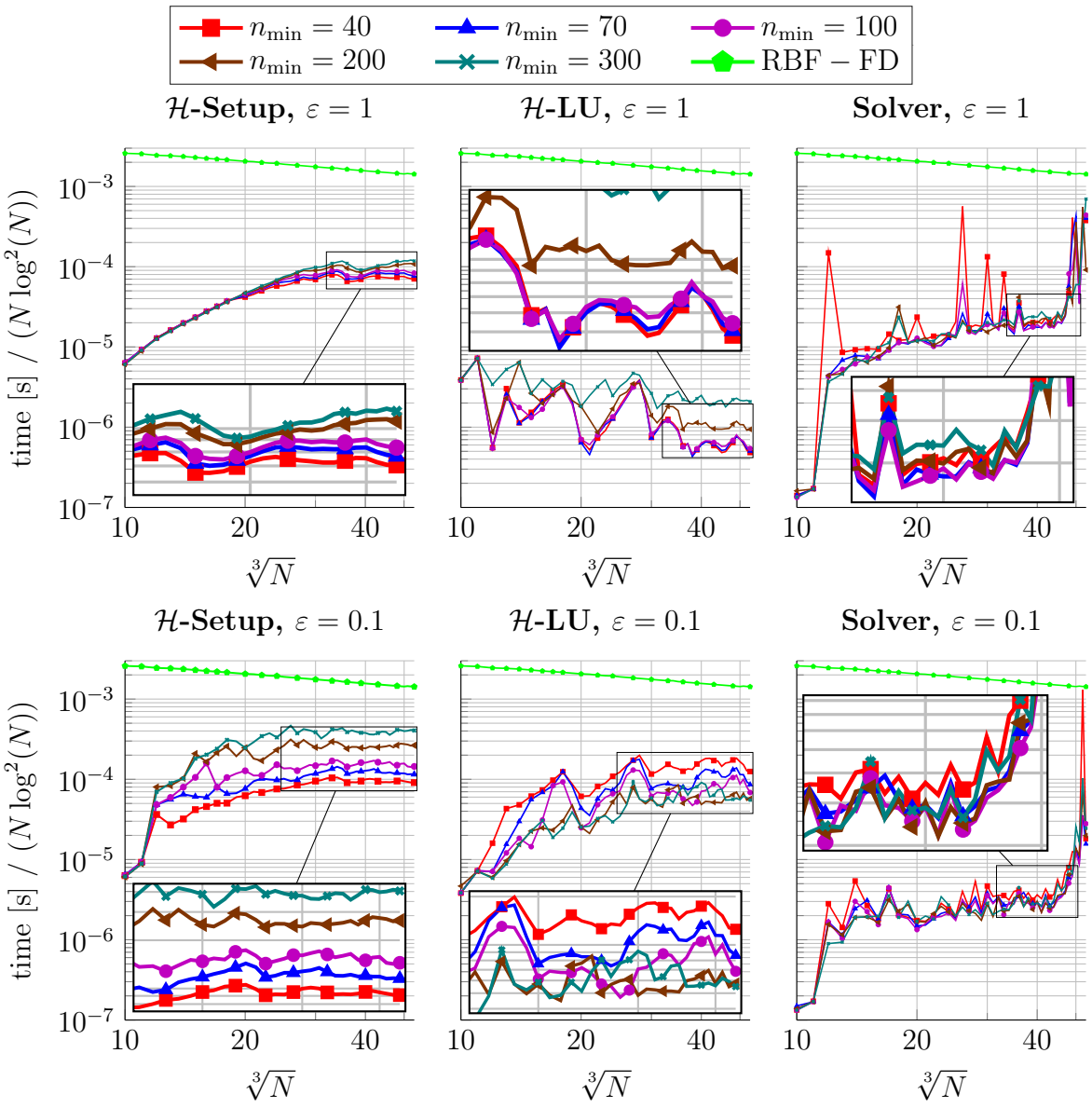

Figure 3.19: Scaled computation times (in seconds) to setup the $\mathcal{H}$-matrix (left) and the $\mathcal{H}$-LU decomposition (middle) and to solve the linear system (right) for PHS($k = 4,\ \ell = 8$) (2.24), PNP nodes, recommended preconditioner setup with $n_{\min} \in \{40, 70, 100, 200, 300\}$ and tolerances $\varepsilon \in \{1, 0.1\}$.

tolerance $\varepsilon = 0.1$ is utilized). Furthermore, the oscillations in the solution time increase with increasing polynomial degree ℓ and no best $n_{\min}$ value can be observed. However, the smallest $n_{\min}$ value leads often to the highest solution times. Nevertheless, the solution times significantly increase for all preconditioner setups at $N \approx 48^3$. The optimal $n_{\min}$ value is larger for larger degrees ℓ, but the concrete $n_{\min}$ value is again of less consequence (than other setup parameters such as the tolerance ε) as long as (3.11) holds. Moreover, the results for tolerance $\varepsilon = 0.1$ indicate the typical behavior

for tolerances $\varepsilon < 1$ that decreasing $n_{\min}$ decreases the setup time of the $\mathcal{H}$-matrix in exchange for an increase in the computation time of the $\mathcal{H}$-LU decomposition (see Figure 3.11). The straighforward way to avoid the case with an empty son in the function build_adaptive_dd_cluster (see Subsection 3.4.4) would be to use a sufficiently large $n_{\min}$. However, a sufficiently large $n_{\min}$ requires $n_{\min} > 400$ for PNP nodes with degree $\ell = 8$ and the results with $n_{\min} \in \{200, 300\}$ demonstrate that a too large $n_{\min}$ can significantly increase the computation time. Hence, we do not recommend to utilize the build_adaptive_dd_cluster function with a sufficiently large $n_{\min}$. An alternative implementation (for a similar idea) would be to add a parameter for the maximal level-number to the function build_adaptive_dd_cluster such that this modified function avoids further partitioning of clusters of higher level.

Figure 3.20 is similar to Figure 3.12 (except that there is an additional "RBF-FD" color/marker as in Figure 3.16) and shows the numbers of iterations and the scaled memory cost of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU factors for the results in Figure 3.19. The left column shows the numbers of iterations and indicates that a larger $n_{\min}$ usually leads to a smaller number of iterations. Furthermore, a too small $n_{\min}$ can significantly increase (the oscillations in) the numbers of iterations, whereas the numbers of iterations significantly increase for all preconditioner setups at $N \approx 48^3$ (which explains the significant increases in the solution times at $N \approx 48^3$ in Figure 3.19). The memory plots in the top row illustrate that for the tolerance $\varepsilon = 1$ a larger $n_{\min}$ leads to larger memory cost of the $\mathcal{H}$-matrix (and its $\mathcal{H}$-LU factors). Moreover, it can be observed again that $\mathcal{H}$-LU preconditioners with coarsening and a high tolerance can lead to preconditioners with lower memory cost than the stiffness matrix. Furthermore, the memory plots in the bottom row demonstrate that, on one hand, the largest memory cost usually corresponds to the largest $n_{\min}$. On the other hand, it can be seen again (as in Figure 3.12) that the smallest memory cost is for a tolerance $\varepsilon < 1$ typically not obtained by the smallest $n_{\min}$ (i. e., $n_{\min} = 100$ leads often to the smallest memory cost).

Figure 3.21 illustrates the approximation errors (2.27) for the RBF-FD discretizations used in Figures 3.19 and 3.20 as a function of the (unscaled) computation time (in seconds, without the setup time of the nodes and the linear system). This demonstrates that an $\mathcal{H}$-LU preconditioner (with a sufficiently small tolerance ε) can be a reliable alternative if an ILU(0) preconditioner fails (see Figure 3.3). However, it can be observed that the approximation errors significantly increase for all preconditioner setups at $N \approx 51^3$ (i. e., roughly the point where the numbers of iterations in Figure 3.20 and the solution times in Figure 3.19 drastically increase). Moreover, a NaN value is computed once for the setup with $\varepsilon = 1$ and $n_{\min} = 40$. Similar observations where RBF-FD discretizations become unstable if a large degree is combined with a large N are also observed in Figure 3.3. Moreover, it can be seen that the smaller the tolerance ε, the larger is the optimal $n_{\min}$. Nevertheless, the lowest computation time often corresponds to a setup with $n_{\min} \geq 100$.

Our tests indicate that the setup time of the $\mathcal{H}$-LU decomposition typically dominates the overall computation time (especially if the linear system should be solved only for

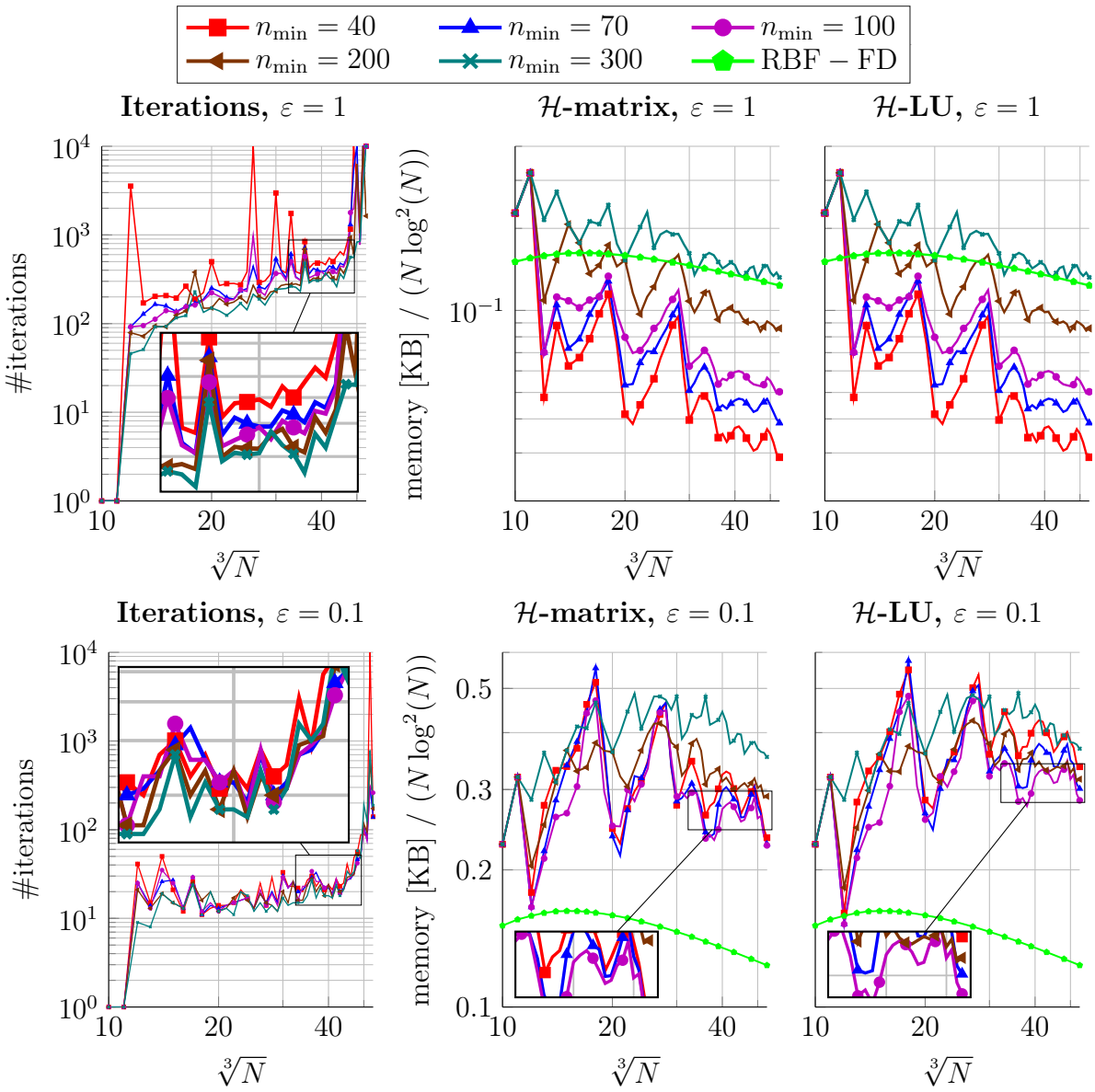

Figure 3.20: Numbers of iterations (left) and scaled memory cost (in KB) for the $\mathcal{H}$-matrix (middle) and the $\mathcal{H}$-LU decomposition (right) for PHS($k = 4$, $\ell = 8$) (2.24), PNP nodes, recommended preconditioner setup with $n_{\min} \in \{40, 70, 100, 200, 300\}$ and tolerances $\varepsilon \in \{1, 0.1\}$.

one right hand side). Hence, we do not show additional tests with coarsening of the $\mathcal{H}$-LU decomposition since this would further increase the setup time of the preconditioner in exchange for a possible (relatively small) reduction in the solution time. Nevertheless, coarsening of the $\mathcal{H}$-LU decomposition can be beneficial in applications which need to solve a linear system for (sufficiently) many right hand sides.

We discussed in Subsection 3.4.2 general results about $\mathcal{H}$-matrices and their usage as

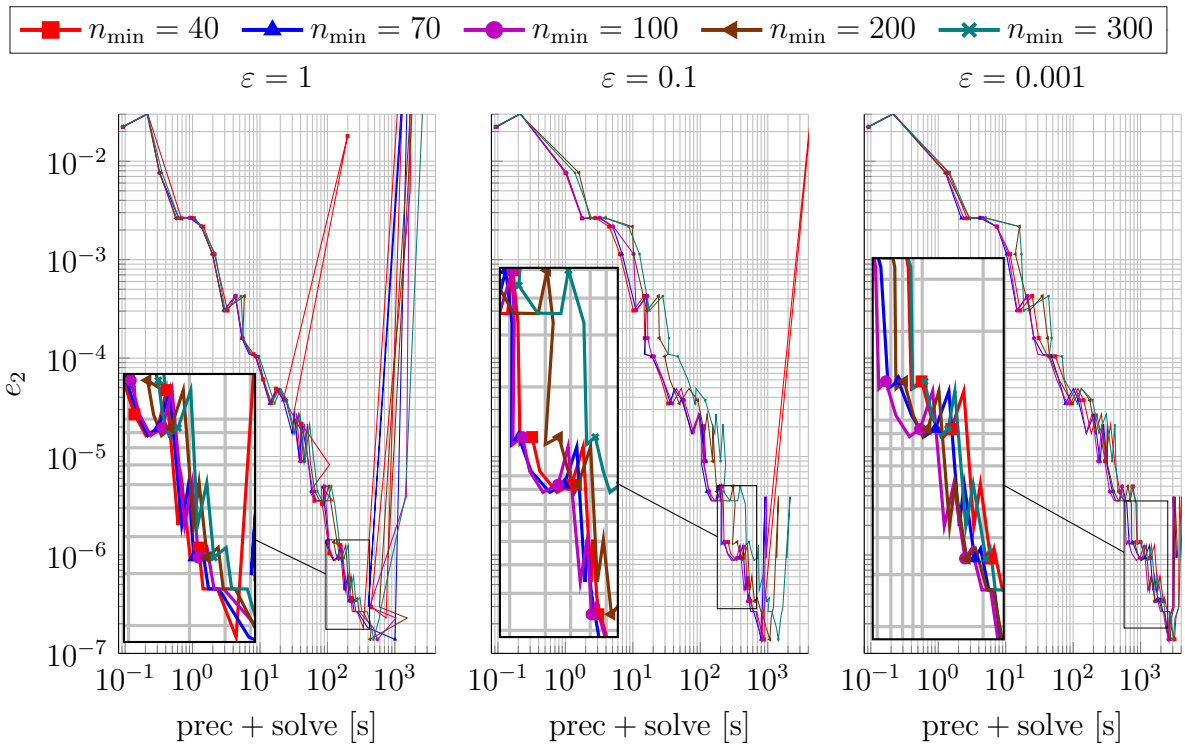

Figure 3.21: Approximation errors (2.27) as a function of the computation time (in seconds, without setup time of the nodes and the linear system) for PHS($k = 4$, $\ell = 8$) (2.24), PNP nodes and recommended preconditioner setup with $n_{\min} \in \{40, 70, 100, 200, 300\}$ and tolerances $\varepsilon \in \{1, 0.1, 0.001\}$.

preconditioners (e. g., as an $\mathcal{H}$-LU preconditioner). Furthermore, we used these general results as well as numerical tests (which are partly included in the previous Subsection 3.4.5 and in this Subsection 3.4.6, whereas the implementation details are given in Subsection 3.4.4) to formulate in Remark 3.20 our recommended $\mathcal{H}$-LU preconditioner setup for an RBF-FD approach. We want to emphasize that our numerical tests revealed the following differences between general sparse matrices (see Subsection 3.4.2) and RBF-FD stiffness matrices. On one hand, coarsening of the $\mathcal{H}$-matrix can significantly reduce the computation time of its $\mathcal{H}$-LU decomposition (and zero full matrix blocks are ideal coarsening candidates). On the other hand, the conversion of zero full matrix blocks to rank-0 matrices in an $\mathcal{H}$-matrix representation can also significantly reduce the computation time as well as the memory cost of its $\mathcal{H}$-LU decomposition (in particular, if no coarsening is used). The main reasons for these differences between RBF-FD and general sparse matrices are that RBF-FD stiffness matrices have a nonsymmetric sparsity structure and often a relatively large stencil size (especially if a large degree of polynomial augmentation is used (2.29)). Furthermore, we summarize our observations about the numerical tests in the previous Subsection 3.4.5 and in this subsection in the following remark.

Remark 3.21. *We observed that the following decisions or parameters significantly influence the overall computation time as well as the memory cost of the $\mathcal{H}$-LU preconditioner.*

- *The admissibility condition (see Definitions 3.9 and 3.10);*
- *the tolerance ε;*
- *the usage of coarsening;*
- *the conversion of zero full matrix blocks to rank-0 matrices.*

Furthermore, we noticed that the following decisions or parameters usually lead to a (relatively) small impact on the overall computation time and the memory cost of the $\mathcal{H}$-LU preconditioner (i. e., these decisions and parameters should be used to consequently fine-tune the $\mathcal{H}$-LU preconditioner).

- *The $n_{\min}$ value (see Definition 3.9, however, a significantly smaller or larger $n_{\min}$ value than* (3.11) *should be avoided);*
- *the usage of accumulated updates [14];*
- *the norm used for truncation accuracy (see Definition 3.14).*

In particular, the comparison between the setups (f) and (h) in Table 3.3 (i. e., the influence of the conversion of zero full matrix blocks to rank-0 matrices if no coarsening is done) illustrated that "false" nonzero entries (see Figure 3.6) can significantly increase the computation and the memory cost of an $\mathcal{H}$-LU preconditioner. Our numerical tests indicate that a weak admissibility condition (instead of an η-admissibility condition) as well as at least one of the approaches coarsening or conversion of zero full matrix blocks to rank-0 matrices should be used if a relative large tolerance ε (such as $\varepsilon \in [0.1, 1]$) is used. Nevertheless, the conversion of zero full matrix blocks to rank-0 matrices is not necessary if coarsening with tolerance $\varepsilon = 1$ (3.15) *is done. Furthermore, the conversion of zero full matrix blocks to rank-0 matrices should be avoided if no coarsening is done and a tolerance $\varepsilon \leq 0.001$ is utilized. Moreover, we observed that accumulated updates [14] frequently lead to higher computation times of the $\mathcal{H}$-LU decomposition than the standard approach, whereas the Frobenius norm can lead to slightly better results (i. e., lower memory cost, lower number of iterations and lower computation time) than the Euclidean norm. Furthermore, the tolerance ε significantly influences both the computation time of an $\mathcal{H}$-LU preconditioner and the (appropriate) choice of the underlying parameters in the $\mathcal{H}$-LU preconditioner setup.*

3.5 Smaller stencil preconditioners

We focus in this section on RBF-FD discretizations with larger degrees $\ell \in \{5, 8\}$ and study preconditioners that are constructed for an auxiliary RBF-FD stiffness matrix with a smaller degree $\ell < 5$. The motivation for this approach is to speedup the setup of the preconditioner since a smaller degree corresponds to a smaller stencil size (see Table

3.1 and (2.29)). We provide the following figures to illustrate the results of this section.

- Figure 3.22: Performance of the ILU(0) preconditioner with degree $\ell = 2$ for discretizations with degrees $\ell \in \{2, 5, 8\}$ and PHS degrees given by (2.24).
- Figure 3.23: Performance of the BiCGstab solver without preconditioner and with ILU(0) preconditioner with degrees $\ell \in \{2, 3, 4, 5\}$ for discretizations with degree $\ell = 5$ and PHS degrees given by (2.24) and (2.26).
- Figure 3.24: Performance of the $\mathcal{H}$-LU preconditioner with degrees $\ell \in \{2, 5\}$ and different tolerances ε for discretizations with degree $\ell = 5$ and PHS degrees given by (2.24) and (2.26).
- Figure 3.25: Performance of the $\mathcal{H}$-LU preconditioner with degrees $\ell \in \{2, 8\}$ and different tolerances ε for discretizations with degree $\ell = 8$ and PHS degrees given by (2.24).
- Figure 3.26: Relationship between the computation time and the approximation errors for the BiCGstab solver without preconditioner as well as the setups from Figures 3.24 and 3.25.

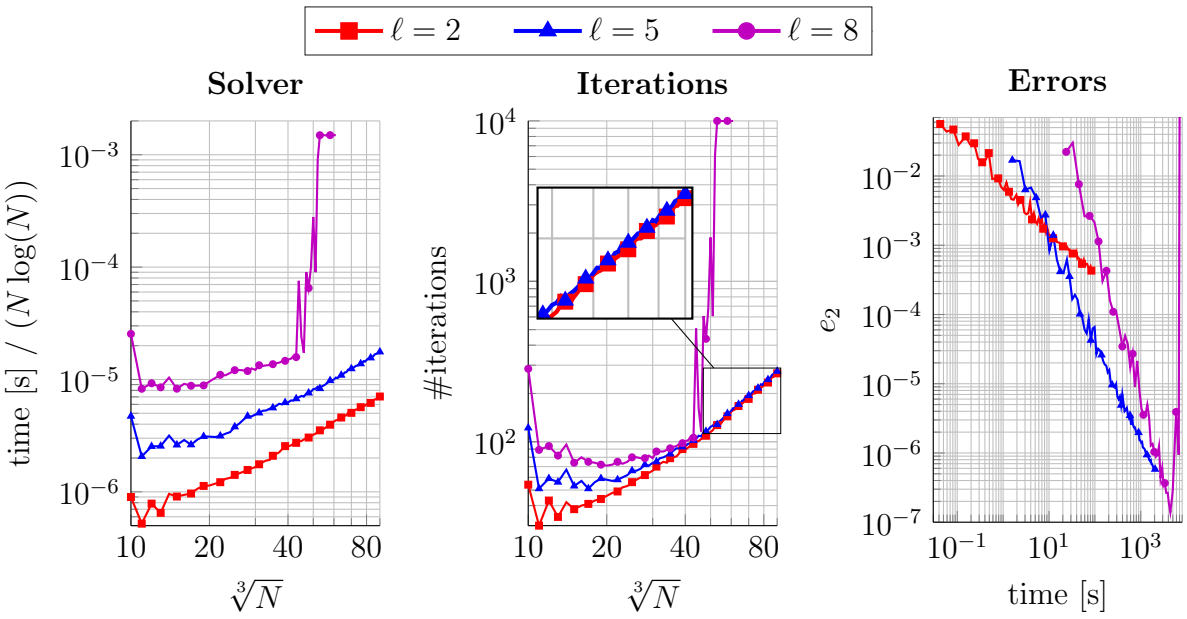

Figure 3.22: Scaled computation times (in seconds) to solve the linear system (left) and numbers of iterations (middle) w. r. t. the number of nodes as well as approximation errors (2.27) (right) w. r. t. the computation time (without setup time of the nodes) for PNP nodes, polynomial degrees $\ell \in \{2, 5, 8\}$, preconditioner with $\ell = 2$ and PHS degrees given by (2.24).

Figure 3.22 illustrates the performance of the ILU(0) preconditioner with degree $\ell = 2$ for discretizations with degrees $\ell \in \{2, 5, 8\}$ and all PHS degrees are given by (2.24). Each color/marker represents a different degree of the discretization. The left plot shows the scaled solution times (in seconds) of the linear system. This plot indicates that the

complexity of the solution of the linear system is similar for the different degrees, where a larger polynomial degree corresponds to a larger constant in the complexity. However, the solution time significantly increases for degree $\ell = 8$ for $N \geq 44^3$. The middle plot shows the numbers of iterations and demonstrates that the gap between the degrees decreases for increasing N, and the numbers of iterations significantly increase for degree $\ell = 8$ for $N \geq 44^3$. We also made in Figures 3.3 and 3.21 the observation that RBF-FD discretizations become unstable if a large degree is combined with a large N. The right plot illustrates the approximation errors (2.27) as a function of the (unscaled) computation time (in seconds, without the setup time of the nodes which is independent of the degree ℓ). This confirms that a higher degree ℓ is advantageous (in terms of the computation time) for high accuracy discretizations, whereas a small ℓ should be preferred for lower accuracies [61]. The results in Figure 3.22 demonstrate that a smaller stencil preconditioner can lead to reliable results (partly even more reliable than an ILU(0) preconditioner for the original linear system, see the orange marker in the last two rows of Figure 3.3).

Figure 3.23 is similar to Figure 3.22 except that we use a fixed degree $\ell = 5$ for the discretization, vary the polynomial degrees $\ell \in \{2, 3, 4, 5\}$ of the (auxiliary) discretizations for which the ILU(0) preconditioner is computed and show in the bottom row in addition results with PHS degrees given by (2.26). Furthermore, we add for comparison results for the plain BiCGstab solver without preconditioner. The red marker in the top row of Figure 3.23 shows the same results as the blue marker in Figure 3.22.

We start with a discussion of the results in the top row. The left plot illustrates that the lowest solution times of the linear system correspond to the polynomial degrees $\ell \in \{3, 4\}$ whereas the plain BiCGstab solver typically needs the largest solution times (except once when the solution for degree $\ell = 5$ does not converge, i. e., see the marker at the left side of the plot). The middle plot shows the numbers of iterations and demonstrates that a larger polynomial degree usually corresponds to a smaller number of iterations as long as the solver converges. However, the differences between the polynomial degrees $\ell \in \{3, 4, 5\}$ are small, whereas the degree $\ell = 2$ leads to significantly larger numbers of iterations. The plain BiCGstab solver typically needs the highest numbers of iterations (except once when the solver for degree $\ell = 5$ does not converge). The right plot illustrates the (unscaled) computation times (in seconds) to setup the preconditioner (and the auxiliary stiffness matrix, if appropriate) and to solve the linear system (i. e., the difference in the computation times between the red marker in this plot and the blue marker in the left plot in Figure 3.22 is the setup time of the linear system). This plot demonstrates that the fastest solution is computed by the setup with degree $\ell = 2$ whereas the slowest solution corresponds to degree $\ell = 4$. This indicates that a smaller stencil preconditioner can speedup (i. e., lead to a faster ILU(0) factorization and faster iteration steps) as well as slow down (i. e., lead to additional setup cost of the auxiliary stiffness matrix and to an usually higher number of iterations) the computation and the choice of the polynomial degree is crucial.

The results in the bottom row (i. e., larger PHS degrees (2.26) instead of (2.24)) demon-

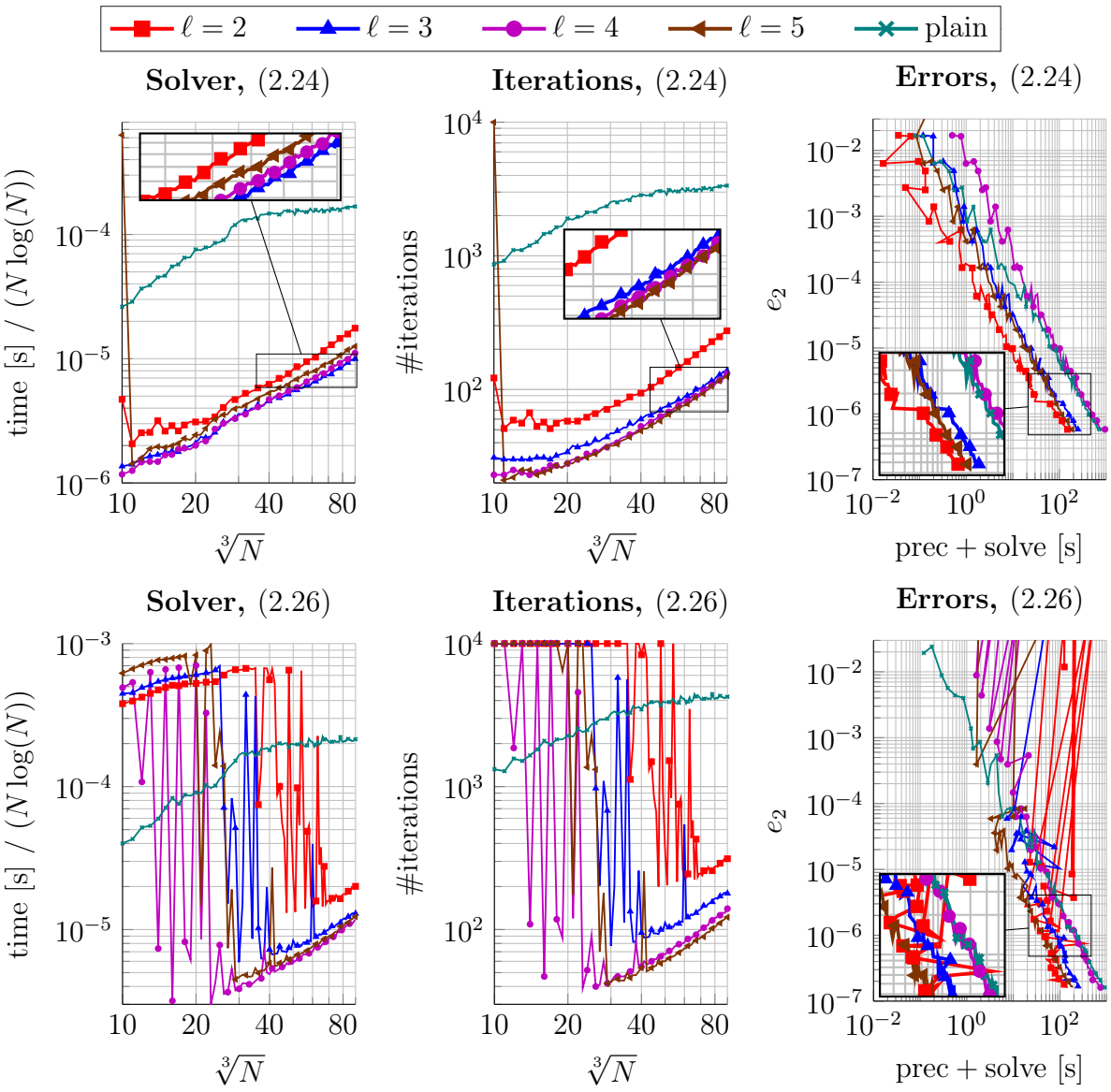

Figure 3.23: Scaled computation times (in seconds) to solve the linear system (left) and numbers of iterations (middle) as functions of the number of nodes as well as approximation errors (2.27) (right) as function of the computation time (without setup time the for the nodes and the linear system) for PNP nodes, polynomial degree $\ell = 5$ and preconditioners with $\ell \in \{2, 3, 4, 5\}$ as well as for the plain BiCGstab solver. PHS degrees via (2.24) (top row) and via (2.26) (bottom row).

strate that the solution times and the numbers of iterations significantly increase for smaller N if larger PHS degrees via (2.26) are used for ILU(0) preconditioners. Furthermore, it can be observed that the BiCGstab solver with an ILU(0) preconditioner often does not converge (and a NaN value is computed once for the preconditioner with de-

gree $\ell = 5$), whereas the plain BiCGstab solver always converges. Moreover, the solution times and the numbers of iterations increase only slightly if a plain BiCGstab solver is utilized. On one hand, the results for ILU(0) preconditioners are not reliable any more for smaller N. Furthermore, smaller stencil preconditioners are not advisable any more since the fastest results usually correspond to the preconditioner for degree $\ell = 5$ or to a plain BiCGstab solver. On the other hand, lower errors (2.27) are possible for larger N and discretizations with larger PHS degrees (2.26) can reach errors below 10^{-6} faster than discretizations with smaller PHS degrees (2.24).

Figure 3.24 illustrates the performance of the $\mathcal{H}$-LU preconditioner with degrees $\ell \in \{2, 5\}$ for discretizations with polynomial degree $\ell = 5$. The PHS degree is determined via (2.24) (top row) and via (2.26) (bottom row). Each color/marker represents a different combination of degree ℓ, tolerance ε and $n_{\min}$. The first five markers correspond to degree $\ell = 2$, tolerances $\varepsilon \in \{1, 0.7, 0.1, 0.01, 0.001\}$ and $n_{\min} = 35$, whereas the last two markers represent degree $\ell = 5$, tolerances $\varepsilon \in \{1, 0.1\}$ and $n_{\min} = 50$ (which are also included as setup (a) in Figures 3.15 and 3.16).

We discuss first the results shown in the first row. The left plot illustrates that for a fixed tolerance ε the setup time of the preconditioner of the auxiliary stiffness matrix (with included setup time of the auxiliary stiffness matrix) is for sufficiently large N significantly lower than the setup time of the preconditioner of the original stiffness matrix (without setup time of the original stiffness matrix). The setup time of the preconditioner of the original linear system with degree $\ell = 5$ and tolerance $\varepsilon = 1$ lies for sufficiently large N between the setup time of the preconditioners with tolerances $\varepsilon \in \{0.7, 0.1\}$ for the auxiliary stiffness matrix with degree $\ell = 2$. The setup time of the preconditioner of the original linear system with degree $\ell = 5$ and tolerance $\varepsilon = 0.1$ lies above the setup time of the preconditioner with tolerance $\varepsilon = 0.01$ (and also above $\varepsilon = 0.001$ except for large N) for the auxiliary stiffness matrix with degree $\ell = 2$.

The middle plot shows that the solution time for the preconditioner with $\ell = 5$ and $\varepsilon = 1$ lies usually between (and once slightly below) the solution times of the preconditioners with $\ell = 2$ and $\varepsilon \in \{1, 0.7\}$. The lowest complexities of the solution time (i. e., approximately $\mathcal{O}(N \log^2(N))$) are obtained for the preconditioners with $\ell = 2$ and $\varepsilon \leq 0.01$, whereas all other preconditioner setups lead to complexities above $\mathcal{O}(N \log^2(N))$. However, the smallest solution time is obtained for smaller N by the preconditioner with $\ell = 5$ and $\varepsilon = 0.1$ whereas the preconditioner with $\ell = 2$ and $\varepsilon = 0.01$ is superior for larger N. Nevertheless, decreasing the tolerance further to $\varepsilon = 0.001$ leads to an increase in the solution time.

The right plot illustrates the numbers of iterations and confirms the observations for the middle plot. The numbers of iterations for the preconditioner with $\ell = 5$ and $\varepsilon = 1$ lie between the numbers of iterations of the preconditioners with $\ell = 2$ and $\varepsilon \in \{1, 0.7\}$. The preconditioners with $\ell = 2$ and $\varepsilon \leq 0.01$ lead to (almost) N independent numbers of iterations (they are even slightly decreasing with increasing N), whereas all other preconditioner setups lead to numbers of iterations which increase with the problem size N. Therefore, the smallest numbers of iterations are obtained for larger N by

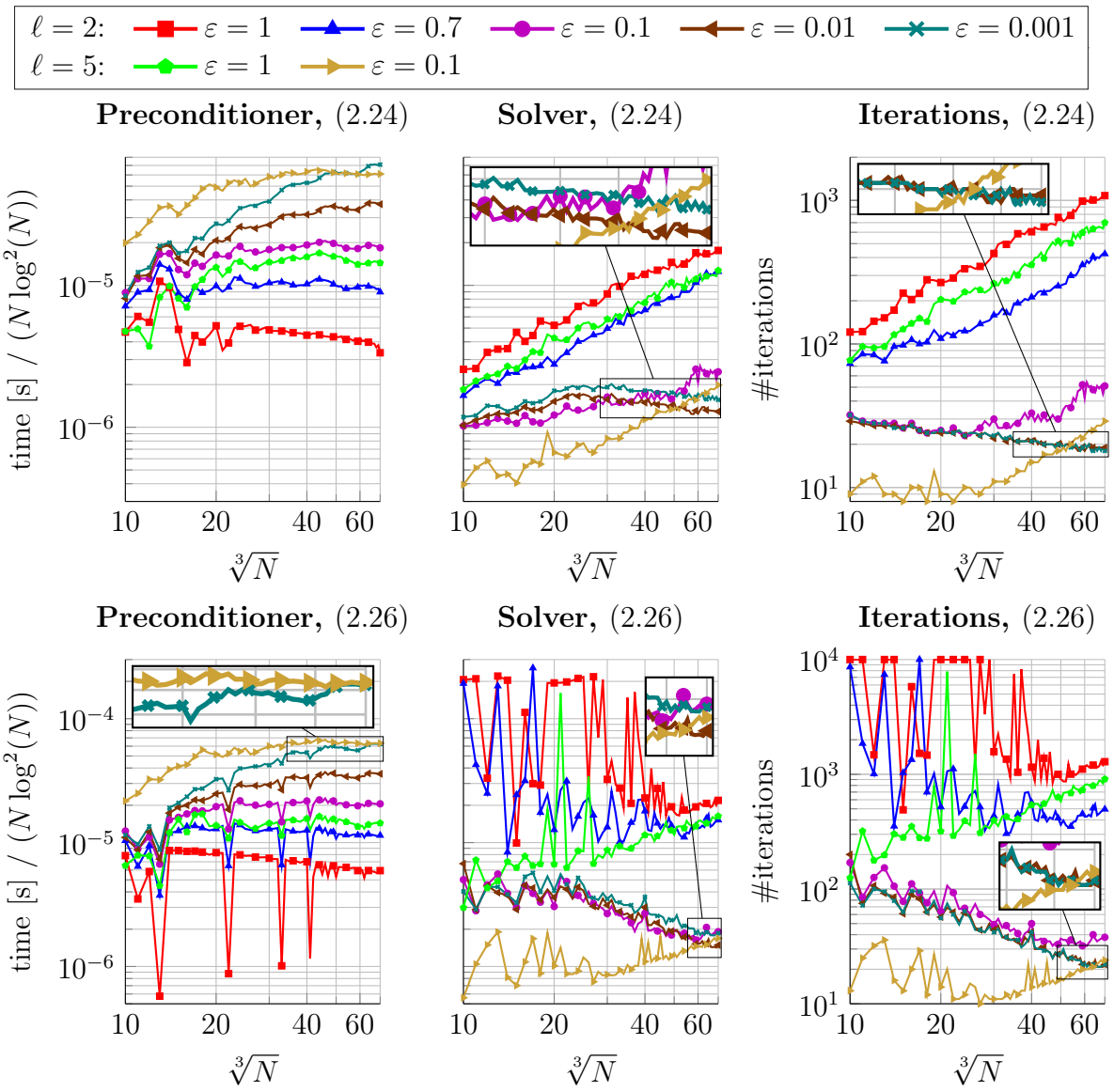

Figure 3.24: Scaled computation times (in seconds) to setup the $\mathcal{H}$-LU preconditioner (left) and to solve the linear system (middle) and numbers of iterations (right) as functions of the number of nodes for PNP nodes, polynomial degree $\ell = 5$ and preconditioners with $\ell = 2$, tolerances $\varepsilon \in \{1, 0.7, 0.1, 0.01, 0.001\}$ and $n_{\min} = 35$ as well as with $\ell = 5$, tolerances $\varepsilon \in \{1, 0.1\}$ and $n_{\min} = 50$. PHS degrees via (2.24) (top row) and via (2.26) (bottom row).

the preconditioners with $\ell = 2$ and $\varepsilon \leq 0.01$, whereas the preconditioner with $\ell = 5$ and $\varepsilon = 0.1$ leads to fewer iterations for smaller N. The numbers of iterations for the preconditioners with $\ell = 2$ and $\varepsilon \in \{0.01, 0.001\}$ are almost identical. The reason for this stagnation in the numbers of iterations is that the auxiliary stiffness matrix is

solely an approximation to the original stiffness matrix (i. e., a further reduction in the number of iterations necessitates a higher degree ℓ for the auxiliary stiffness matrix). This demonstrates that the tolerance ε does not need to be too small if a smaller stencil preconditioner is utilized.

The results in the bottom row illustrate again that a higher PHS degree can significantly increase the solution time and the number of iterations (i. e., lead to numerical instabilities) whereas its influence on the setup time of the preconditioner is negligible. Furthermore, the solvers with smaller stencil preconditioners with tolerances $\varepsilon \geq 0.7$ can fail to converge. The outliers in the setup time of the preconditioners (especially visible for the case with $\ell = 2$ and $\varepsilon = 1$) originate from the method of computing the setup time of the auxiliary stiffness matrix and the structure _clustergeometry (see Subsection 3.4.4) for its $\mathcal{H}$-matrix representation. Our implementation combines the setups of the linear system, the auxiliary stiffness matrix and the structure _clustergeometry to perform only once the setup of the kd-tree and the neighbor search (instead of two or three times). On one hand, this avoids unnecessary repetitions of already performed computations. On the other hand, this interweaves partly the setups of the stiffness matrix, the auxiliary stiffness matrix and the structure _clustergeometry. Hence, this makes the time measurement more difficult (i. e., to evaluate how much computation time is actually spend either for the stiffness matrix or the auxiliary stiffness matrix and the structure _clustergeometry). Therefore, we measure the combined computation time of the auxiliary stiffness matrix and the structure _clustergeometry via the difference between the computation times of two individual computations, namely the combined setup (i. e., original linear system, auxiliary stiffness matrix and structure _clustergeometry) and the setup of solely the original linear system (i. e., without the auxiliary stiffness matrix and structure _clustergeometry). Theoretically, we would expect that the computation time of the combined setup time is larger than the computation time of the setup of solely the original linear system (since the former computation is the latter computation enhanced by some additional computations). However, our tests reveal that the difference between these computation times is negligible and it can occur in practice that the computation time of the combined setup time is smaller than the computation time of the setup of solely the original linear system (i. e., the difference between different runs of the setup of the original linear system is larger than the additional time for the setup of the auxiliary stiffness matrix and the structure _clustergeometry). Therefore, we utilize (3.16) to determine in our plots the setup time of the auxiliary stiffness matrix and the structure _clustergeometry

$$\max\{\text{time(combined setup)} - \text{time(solely original linear system)},\ 0\}. \tag{3.16}$$

On one side, the zero values resulting from (3.16) can lead to visible outliers (especially for the case with $\ell = 2$ and $\varepsilon = 1$) since the setup time of the $\mathcal{H}$-LU preconditioner is in this case relatively small (see fifth row in Figure 3.8). On the other side, this highlights that the additional computation time to setup the auxiliary stiffness matrix and the structure _clustergeometry is negligible if the polynomial degree of augmentation of the original linear system is significantly larger than the polynomial degree of augmentation

of the auxiliary linear system. Nevertheless, the outliers lie in a vertical line (i. e., they are independent of the tolerance ε since they originate from (3.16)).

All in all, it can be observed that a too high accuracy of the preconditioner of the auxiliary system is not needed (e. g., a decrease in the tolerance from $\varepsilon = 0.01$ to $\varepsilon = 0.001$ mainly increases the setup time of the preconditioner and the solution time whereas the number of iterations is almost unchanged). The optimal tolerance for a smaller stencil preconditioner seems to be in (or close to) $[0.01, 0.1]$.

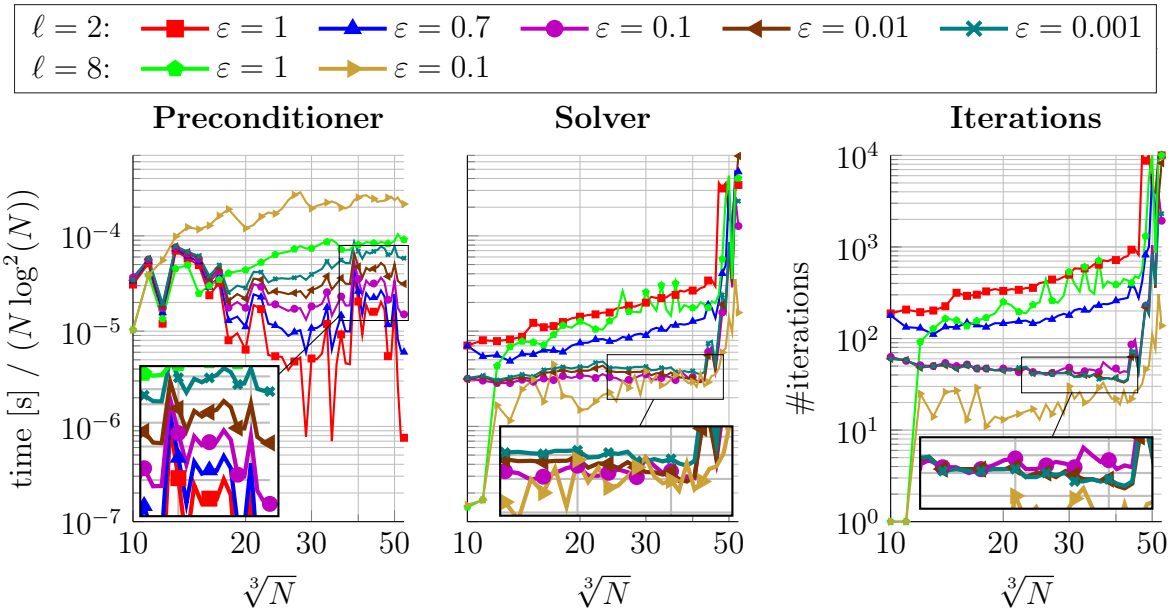

Figure 3.25: Scaled computation times (in seconds) to setup the $\mathcal{H}$-LU preconditioner (left) and to solve the linear system (middle) and numbers of iterations (right) as functions of the number of nodes for PNP nodes, polynomial degree $\ell = 8$ and preconditioners with $\ell = 2$, tolerances $\varepsilon \in \{1, 0.7, 0.1, 0.01, 0.001\}$ and $n_{\min} = 35$ and with $\ell = 8$, tolerances $\varepsilon \in \{1, 0.1\}$ and $n_{\min} = 70$. PHS degrees via (2.24).

Figure 3.25 is similar to Figure 3.24 with discretizations with degree $\ell = 8$ instead of $\ell = 5$, PHS degrees only via (2.24) (and changed axes). The oscillations and outliers in the setup time of the preconditioners (especially visible for the case with $\ell = 2$ and $\varepsilon = 1$) originate again from the method of computing the setup time of the auxiliary stiffness matrix (see the discussion to Figure 3.24, in particular (3.16)). The observations for Figure 3.25 are qualitatively similar to the observations for Figure 3.24 and the optimal tolerance seems to be again approximately $\varepsilon = 0.1$. The main difference is that the solution times and the numbers of iterations significantly increase for large problem sizes (with $N \approx 48^3$ and larger). These observations are rather problems of the discretization than of the preconditioner (see Figure 3.21).

Figure 3.26 combines the cases tested in Figures 3.24 and 3.25 (without the tolerances $\varepsilon \in \{0.7, 0.001\}$ for better distinguishability), contains in addition results for the plain

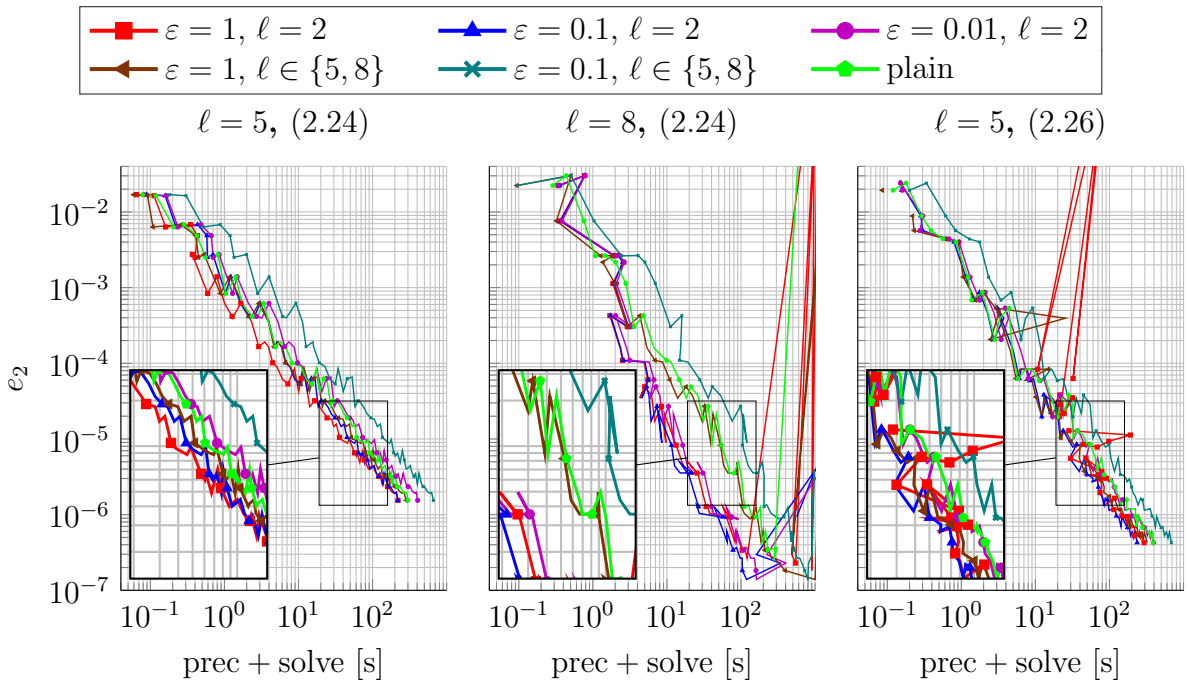

Figure 3.26: Approximation errors (2.27) as a function of the computation time (in seconds, without setup time of the nodes and the linear system) for polynomial degrees $\ell \in \{5, 8\}$, PNP nodes and for the plain BiCGstab solver as well as with the recommended preconditioner setup with $n_{\min} \in \{35, 50, 70\}$ and tolerances $\varepsilon \in \{1, 0.1, 0.01\}$. PHS degrees via (2.24) and (2.26).

BiCGstab solver without preconditioner and shows the corresponding approximation errors (2.27). This figure is similar to Figures 3.17 and 3.21 (i. e., the time is given by the accumulated times shown in the first two columns of Figures 3.24 and 3.25) and confirms that smaller stencil preconditioners can lead to fast and reliable results. The fastest results are often obtained by the preconditioner for degree $\ell = 2$ with tolerance $\varepsilon = 0.1$ whereas the slowest results usually correspond to the preconditioner of the original stiffness matrix with tolerance $\varepsilon = 0.1$. Additionally, this figure illustrates once again that all tested solvers and preconditioners (i. e., ILU(0) with LPBiCGstab, LPGMRES and RPGMRES in Figure 3.3, $\mathcal{H}$-LU in Figure 3.21 and smaller stencil preconditioners in Figures 3.22 and 3.26) lead to numerical instabilities for polynomial degree $\ell = 8$ on discretizations with approximately $N \geq 50^3$. This indicates that rather the RBF-FD discretization (for this parameter choices) than the solver setup becomes numerically instable (since we observed that $\mathcal{H}$-LU preconditioners with a sufficiently small tolerance ε can cope well with more "challenging" RBF-FD discretizations such as for Halton nodes or larger PHS degrees (2.26)). However, the accuracy barrier in double precision w. r. t. floating point arithmetic (2.17) is not reached and shifting the stencils (see Remark 2.7) does not solve this problem (since similar observations are reported in [60] which uses shifted stencils in the Medusa library [112]). Hence, the condition numbers of the underlying (stiffness) matrices B could be studied further, i. e., whether

they fulfill $\mathcal{K}(B) \geq 10^9$ (2.16).

Furthermore, we performed tests (not included here) in which we solved the auxiliary linear system first and used this solution as the start vector for the original linear system. These tests indicated that this changed start vector can decrease as well as increase the numbers of iterations. However, these changes in the numbers of iterations were often negligible whereas the additional solution time for the auxiliary linear system (and the setup time of the right hand side of the auxiliary linear system) increased the overall computation time. Therefore, we do not study this approach here further.

3.6 Summary, conclusion and outlook

Our numerical tests focused in this chapter on the convection diffusion problem with the diffusion coefficient $\nu = 10^{-3}$ and a recirculating convection (i. e., on the differential operator $\mathcal{L} = -\nu\Delta + b^T\nabla$ with a recirculating convective flow b). Further tests (not included here) indicated that our qualitative results also hold for larger diffusion coefficients and the Poisson problem. The main difference between these tests for larger diffusion coefficients and the Poisson problem not shown and our tests in this section (for the diffusion coefficient $\nu = 10^{-3}$) are that the performance of ILU(0) preconditioners and $\mathcal{H}$-LU preconditioners with a large tolerance ε (e. g., with $\varepsilon = 1$) is decreased for the case with diffusion coefficient $\nu = 10^{-3}$ (i. e., almost singular ILU(0) decompositions, oscillations in the numbers of iterations and cases with divergence in the iterative solver occur more often than for the cases with $\nu > 10^{-3}$).

BiCGstab equipped with an ILU(0) preconditioner (or another blackbox solver such as for example an algebraic multigrid solver or a sparse direct solver) is a straighforward (and in the literature often used) option to solve the sparse linear system in the RBF-FD method. Nevertheless it has been observed that the solution of this linear system can be the bottleneck of the whole computation (especially if the problem size is large). Hence, fast and reliable solvers (and typically problem dependent preconditioners) are important. Therefore, we illustrated the advantages and disadvantages of the solvers BiCGstab and GMRES, of left and right preconditioning and of ILU(0) preconditioners in Section 3.3. Our observations are that

- BiCGstab typically leads to similar or smaller solution times than GMRES (as long as BiCGstab converges);
- the difference between left and right preconditioning is negligible if a very small relative residual tolerance is used for the iterative solver.

Furthermore, we demonstrated that BiCGstab combined with an ILU(0) preconditioner can be a fast and reliable solver, especially if

- the degree ℓ (or here equivalently the stencil size n) is small;
- the PHS degree is determined via (2.24);

- the linear system should be solved for only one right hand side;
- the node distribution is more regular (e. g., Cartesian or PNP nodes);
- the problem size N is small.

In particular, a small condition of the underlying linear system (which is more likely for more regular node distributions and smaller problem sizes N) is beneficial for the performance of an iterative solver.

The results in Section 3.3 illustrate that the combination of BiCGstab with ILU(0) preconditioner can be a fast and reliable solver setup, however problems can occur for large polynomial degrees, large PHS degrees (2.26) and unstructured nodes (which are the strength of the RBF-FD approach). Therefore, we concentrated in Section 3.4 on an alternative solver setup, namely on $\mathcal{H}$-matrices and their application in the solution of linear systems (especially the linear system in the RBF-FD approach). We illustrated that $\mathcal{H}$-LU preconditioners (combined with a nested dissection ordering) can be a promising alternative to an ILU(0) preconditioner, especially if

- the computation of the ILU(0) preconditioner failed;
- the solver with the ILU(0) preconditioner does not converge;
- the degree ℓ (or here equivalently the stencil size n) is large;
- the PHS degree is determined via (2.26);
- the linear system should be solved for several right hand sides;
- the node distribution is more irregular;
- the condition of the underlying linear system is large;
- the problem size N is large.

However, the setup of an $\mathcal{H}$-LU preconditioner can be expensive and its cost is significantly influenced by the choice of the tolerance ε and the admissibility condition (see Definition 3.9). Furthermore, we observed that the weak and the η-admissibility conditions (see Definition 3.10) typically lead to preconditioners with a comparable quality, whereas the cost can be significantly reduced for the weak admissibility condition (which eliminates in addition the parameter η). Moreover, we illustrated that our $\mathcal{H}$-LU preconditioner setup (in particular, its implementation, see Subsection 3.4.4) would benefit from a symmetric sparsity structure of the stiffness matrix (see Figure 3.6). Hence, it could be beneficial to use a different approach than nearest neighbor search to setup the stencils, namely to select a fixed radius and use all nodes within this radius w. r. t. the stencil center [85] (i. e., different stencil sizes n_j for different stencils X_j may occur). In particular, regular node distributions (such as Cartesian nodes) should rely as often as possible on symmetric stencils (e. g., to use a stencil size $n \in \{19,\ 27\}$ instead of $n = 22$ (2.29) for Cartesian nodes with polynomial augmentation of degree $\ell = 2$ if no unsymmetric stencil is needed due to upwinding). For a practical application of an $\mathcal{H}$-LU preconditioner (in the context of RBF-FD), we recommend to consider Remarks 3.20 and 3.21 as well as the following approach.

- Select first the (desired or needed) truncation tolerance ε (3.7) (for simplicity, use the same tolerance ε for all $\mathcal{H}$-matrix arithmetic)).
- If ε is relative high (e. g., $\varepsilon \geq 0.001$), the use of coarsening, the weak admissibility condition and the conversion of zero full matrix blocks to rank-0 blocks can be promising choices. However, these choices can be suboptimal and not advisable if the tolerance ε is relative small (e. g., $\varepsilon < 0.001$).
- Use $n_{\min}$ according to (3.11) with the rule of thumb: the higher the tolerance ε, the smaller should be $n_{\min}$.
- Use the relative error in the Frobenius (or in the Euclidean) norm and the standard $\mathcal{H}$-LU decomposition (i. e., avoid accumulated updates).

The results in Section 3.4 indicate that $\mathcal{H}$-LU preconditioners can be a promising alternative to an ILU(0) preconditioner. However, their major drawback is their usually large setup cost. Therefore, we introduced and tested in Section 3.5 another option to speedup the setup of the preconditioner, namely the construction of a preconditioner for an auxiliary stiffness matrix with a smaller stencil size. We demonstrated that these preconditioners can be superior to preconditioners of the original linear system since their setup time can be significantly reduced (even if the additional setup cost of the auxiliary stiffness matrix is included), whereas their quality is often comparable (and they can be even more reliable than a preconditioner for the original stiffness matrix). This relatively low setup cost of a smaller stencil preconditioner can be especially beneficial, if the linear system should be solved for only one right hand side since a high accuracy $\mathcal{H}$-LU preconditioner for an auxiliary stiffness matrix with a smaller stencil size can still lead to problem size independent number of iterations. The stencil size of the auxiliary stiffness matrix can be seen as an additional parameter for problem dependent fine-tuning (i. e., a larger stencil size typically increases the setup cost in exchange for a decreased number of iterations, whereas a too small stencil size could hinder the convergence of the solver).

Our novel contributions of this chapter consist of

- numerical tests and recommendations on the development of hierarchical matrix preconditioners for RBF-FD sparse matrices;
- the introduction of preconditioners that are based on an auxiliary RBF-FD stiffness matrix with a smaller stencil size.

Hence, this work can be seen, on one hand, as a proof of concept that smaller stencil preconditioners can be a promising option. On the other hand, we stated guidelines for the construction and application of $\mathcal{H}$-LU preconditioners in the context of the RBF-FD method. However, the optimal choice of the involved parameters is problem dependent.

One straightforward extension of the work in this chapter (and also of the work in the previous chapter, see Section 2.5) would lie in the consideration of further RBF-FD variants, including stabilized versions, variable shape parameters, other than nearest neighbor stencils, further types of generating functions or their combinations or other

types of partial differential equations. Furthermore, the limitations of RBF-FD w. r. t. larger degrees of polynomial augmentation and the resulting occurrence of numerical instabilities observed in Chapter 3 could be further studied. The work in this chapter can be seen as a proof of concept to illustrate that $\mathcal{H}$-LU preconditioners (optionally for an auxiliary stiffness matrix with a smaller stencil size) can be a promising option for the solution of the linear system in the RBF-FD method. Hence, other extensions of the work in this chapter would be on one side to focus on theoretical results, e. g., about the underlying computation time and memory complexities. On the other side, more sophisticated ideas (see Subsection 3.4.4) such as

- further coarsening approaches or their combination,
- the inclusion of the information given by the kd-tree or obtained during the stencil selection to construct the (block) cluster tree or the admissibility condition (e. g., with consideration to the nonsymmetric sparsity structure of the stiffness matrix),
- an adaptive or alternating construction of the stiffness matrix and its $\mathcal{H}$-matrix representation

could be tested. Further extensions could be the fine-tuning of the recommendations for the multitude of (RBF-FD and preconditioner) parameters or their determination via machine learning or auto-tuning.

4 Conclusion and outlook

Chapter 1 started with a motivation for the numerical solution of PDEs via the RBF-FD approach. The main advantages of the RBF-FD approach are: it is meshfree and its application is (relatively) straighforward.

We discussed basic but often still competitive versions of the RBF-FD approach in Chapter 2 and illustrated that several methods work well if parameters are chosen according to our recommendations in Subsection 2.4.7. However, we demonstrated in addition the typical and well known pitfalls which can arise in RBF-FD discretizations if parameters are not selected carefully. Our novel contributions of this chapter consisted of

- a comprehensive view and comparison of basic RBF-FD versions from the literature;
- general recommendations for parameter setups in basic RBF-FD methods such as a general recommendation to shift stencils to the origin if polynomials of large degrees are augmented and a new scaling law for the stencil size and the degree of polynomial augmentation on Cartesian node distributions.

Afterwards, we concentrated in Chapter 3 on the solution of the global linear system of equations which is the bottleneck of the whole simulation for large problem sizes. We demonstrated the advantages and disadvantages of ILU(0) preconditioners and illustrated in a proof of concept approach via numerical examples that $\mathcal{H}$-LU preconditioners as well as preconditioners which are constructed for an auxiliary RBF-FD stiffness matrix with a smaller stencil size can be promising options for the solution of the global linear system of equations. On one hand, $\mathcal{H}$-LU preconditioners are especially attractive if other preconditioners (e. g., ILU(0)) lead to divergence of the iterative solver, or if the linear system should be solved for several right hand sides. On the other hand, a smaller stencil size preconditioner based on an $\mathcal{H}$-LU decomposition is a reasonable compromise to obtain a reliable preconditioner with a relatively small setup time (i. e., compared to an $\mathcal{H}$-LU preconditioner of the original discretization with a larger stencil size). Our novel contributions of this chapter consisted of

- numerical tests and recommendations on the development of hierarchical matrix preconditioners to RBF-FD sparse matrices;
- the introduction of preconditioners that are based on an auxiliary RBF-FD stiffness matrix with a smaller stencil size.

There are many further research directions such as more complicated test problems, the development and (theoretical) study of more advanced RBF-FD versions, the application

of machine learning or auto-tuning to determine the setup parameters. Many of these research activities can benefit from existing software libraries such as the Medusa library [112] for the RBF-FD method and the $\mathcal{H}$-matrix library H2Lib [52] for efficient handling of matrix arithmetic.

Furthermore, the research on the application of $\mathcal{H}$-matrices in the RBF-FD approach can be extended. More sophisticated coarsening approaches which conduct coarsening of the cluster tree in a preprocessing step and coarsening of the block cluster tree in a postprocessing step can be investigated. The exchange of geometric information between the setup of the stiffness matrix and its $\mathcal{H}$-matrix representation can be studied. Is it beneficial and feasible to include more geometric information of the kd-tree which is computed in the setup of the stiffness matrix in the computation of the (block) cluster tree (e. g., to include geometric information of the kd-tree in the nested dissection ordering)? Is an adaptive or alternating construction of the stiffness matrix and its $\mathcal{H}$-matrix representation (see Subsection 3.4.4) reasonable and can the approximation accuracy of the original RBF-FD approximation be preserved? Another idea is to either create a symmetric sparsity structure of the stiffness matrix or to take the in general nonsymmetric sparsity structure of the stiffness matrix into consideration (i. e., to focus on the development of block cluster trees that are based on distinct cluster trees for the rows and the columns). This approach could reduce the amount of full matrix blocks and increase the size of zero blocks in the $\mathcal{H}$-matrix representation such that memory cost of the $\mathcal{H}$-matrix and its $\mathcal{H}$-LU decomposition is decreased and the computations are accelerated. Additionally, the treatment of large full matrix blocks (which typically do not occur in the $\mathcal{H}$-matrix representation of FD and FE stiffness matrices) that cannot be further divided can be investigated, e. g., is a low-rank approximation in some cases more efficient than a full matrix representation?

Another direction for further research are preconditioners that are constructed for an auxiliary stiffness matrix with a smaller stencil size. Are they still reliable for significantly larger and more challenging problems? How should this smaller stencil size be determined to obtain problem size independent numbers of iterations of the iterative solver?

References

[1] G. Arora et al. "A review of radial basis function with applications explored". *J. Egyptian Math. Soc.* 31, 6 (2023).

[2] N. B. Barik and T. V. S. Sekhar. "Multilevel Meshfree RBF-FD Method for Elliptic Partial Differential Equations". In: *Engineering Mathematics and Computing*. Springer, 2023, pp. 1–10.

[3] N. Bartwal et al. "Application of a High Order Accurate Meshless Method to Solution of Heat Conduction in Complex Geometries". *arXiv:2106.08535* (2021).

[4] N. Bartwal et al. "Simulation of heat conduction in complex domains of multi-material composites using a meshless method". *Appl. Math. Comput.* 457, 128208 (2023).

[5] B. Baxter and R. Brummelhuis. "Convergence of stationary radial basis function-schemes for evolution equations". *arXiv:1905.01128* (2019).

[6] V. Bayona. "An insight into RBF-FD approximations augmented with polynomials". *Comput. Math. Appl.* 77 (2019), pp. 2337–2353.

[7] V. Bayona, N. Flyer, and B. Fornberg. "On the role of polynomials in RBF-FD approximations: III. Behavior near domain boundaries". *J. Comput. Phys.* 380 (2019), pp. 378–399.

[8] V. Bayona, M. Moscoso, and M. Kindelan. "Gaussian RBF-FD weights and its corresponding local truncation errors". *Eng. Anal. Bound. Elem.* 36 (2012), pp. 1361–1369.

[9] V. Bayona et al. "A 3-D RBF-FD solver for modeling the atmospheric global electric circuit with topography (GEC-RBFFD v1. 0)". *Geoscientific Model Development* 8 (2015), pp. 3007–3020.

[10] V. Bayona et al. "On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs". *J. Comput. Phys.* 332 (2017), pp. 257–273.

[11] V. Bayona et al. "RBF-FD formulas and convergence properties". *J. Comput. Phys.* 229 (2010), pp. 8281–8295.

[12] M. Benzi and A. J. Wathen. "Some Preconditioning Techniques for Saddle Point Problems". In: *Model Order Reduction: Theory, Research Aspects and Applications*. Vol. 13. Math. Ind. 2008, pp. 195–211.

[13] E. F. Bollig, N. Flyer, and G. Erlebacher. "Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs". *J. Comput. Phys.* 231 (2012), pp. 7133–7151.

[14] S. Börm. "Hierarchical matrix arithmetic with accumulated updates". *Comput. Vis. Sci.* 20 (2019), pp. 71–84.

[15] J. P. Boyd and K. W. Gildersleeve. "Numerical experiments on the condition number of the interpolation matrices for radial basis functions". *Appl. Numer. Math.* 61 (2011), pp. 443–459.

[16] M. Bozzini et al. "Interpolation with variably scaled kernels". *IMA J. Numer. Anal.* 35 (2015), pp. 199–219.

[17] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge Monogr. Appl. Comput. Math. 2003.

[18] J. Burkardt. *HALTON: The Halton Quasi Monte Carlo (QMC) Sequence*. `https://people.sc.fsu.edu/~jburkardt/c_src/halton/halton.html`. 2016. Retrieved Juli 25, 2018.

[19] Y. L. Chan et al. "A novel upwind-based local radial basis function differential quadrature method for convection-dominated flows". *Comput. & Fluids* 89 (2014), pp. 157–166.

[20] P. Chinchapatnam et al. "A compact RBF-FD based meshless method for the incompressible Navier-Stokes equations". *Proceedings of The Institution of Mechanical Engineers Part M-journal of Engineering for The Maritime Environment* 223 (2009), pp. 275–290.

[21] O. Davydov. "Approximation with Conditionally Positive Definite Kernels on Deficient Sets". In: *Approximation Theory XVI: Nashville 2019*. Springer. 2021, pp. 27–38.

[22] O. Davydov. "Selection of Sparse Sets of Influence for Meshless Finite Difference methods". *arXiv:1908.01567* (2019).

[23] O. Davydov and D. T. Oanh. "Adaptive meshless centres and RBF stencils for Poisson equation". *J. Comput. Phys.* 230 (2011), pp. 287–304.

[24] O. Davydov, D. T. Oanh, and N. M. Tuong. "Improved stencil selection for meshless finite difference methods in 3D". *J. Comput. Appl. Math.* 425, 115031 (2023).

[25] O. Davydov and R. Schaback. "Minimal numerical differentiation formulas". *Numer. Math.* 140 (2018), pp. 555–592.

[26] O. Davydov and R. Schaback. "Optimal stencils in Sobolev spaces". *IMA J. Numer. Anal.* 39 (2017), pp. 398–422.

[27] M. Depolli, J. Slak, and G. Kosec. "Parallel domain discretization algorithm for RBF-FD and other meshless numerical methods for solving PDEs". *Computers & Structures* 264, 106773 (2022).

[28] T. Dobravec, B. Mavrič, and B. Šarler. "Acceleration of RBF-FD meshless phase-field modelling of dendritic solidification by space-time adaptive approach". *Comput. Math. Appl.* 126 (2022), pp. 77–99.

[29] T. A. Driscoll and B. Fornberg. "Interpolation in the limit of increasingly flat radial basis functions". *Comput. Math. Appl.* 43 (2002), pp. 413–422.

[30] U. Duh, G. Kosec, and J. Slak. "Fast variable density node generation on parametric surfaces with application to mesh-free methods". *SIAM J. Sci. Comput.* 43 (2021), A980–A1000.

[31] U. Duh, G. Kosec, and J. Slak. *Standalone implementation of the proposed algorithm*. `http://e6.ijs.si/medusa/static/PA.zip`. Retrieved January 11, 2022.

[32] U. Duh, V. Shankar, and G. Kosec. "Discretization of non-uniform rational B-spline (NURBS) models for meshless isogeometric analysis". *arXiv:2303.02638* (2023).

[33] G. E. Fasshauer. *Meshfree approximation methods with MATLAB*. Vol. 6. Interdiscip. Math. Sci. 2007.

[34] G. E. Fasshauer. "Positive Definite Kernels: Past, Present and Future". *Dolomite Res. Notes Approx.* 4 (2011), pp. 21–63.

[35] G. E. Fasshauer and M. J. McCourt. *Kernel-based Approximation Methods using MATLAB*. Vol. 19. Interdiscip. Math. Sci. 2015.

[36] G. E. Fasshauer and J. G. Zhang. "On choosing "optimal" shape parameters for RBF approximation". *Numer. Algorithms* 45 (2007), pp. 345–368.

[37] H. Faure, P. Kritzer, and F. Pillichshammer. "From van der Corput to modern constructions of sequences for quasi-Monte Carlo rules". *Indag. Math. (N.S.)* 26 (2015), pp. 760–822.

[38] N. Flyer, G. A. Barnett, and L. J. Wicker. "Enhancing finite differences with radial basis functions: Experiments on the Navier-Stokes equations". *J. Comput. Phys.* 316 (2016), pp. 39–62.

[39] N. Flyer, G. B. Wright, and B. Fornberg. "Radial Basis Function-Generated Finite Differences: A Mesh-Free Method for Computational Geosciences". In: *Handbook of Geomathematics*. Springer Berlin Heidelberg, 2014, pp. 1–30.

[40] N. Flyer et al. "A guide to RBF-generated finite differences for nonlinear transport: shallow water simulations on a sphere". *J. Comput. Phys.* 231 (2012), pp. 4078–4095.

[41] N. Flyer et al. “On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy”. *J. Comput. Phys.* 321 (2016), pp. 21–38.

[42] B. Fornberg and N. Flyer. *A Primer on Radial Basis Functions with Applications to the Geosciences*. Vol. 87. CBMS-NSF Regional Conf. Ser. in Appl. Math. 2015.

[43] B. Fornberg and N. Flyer. “Solving PDEs with radial basis functions”. *Acta Numer.* 24 (2015), pp. 215–258.

[44] B. Fornberg, E. Lehto, and C. Powell. “Stable calculation of Gaussian-based RBF-FD stencils”. *Comput. Math. Appl.* 65 (2013), pp. 627–637.

[45] B. Fornberg, G. Wright, and E. Larsson. “Some observations regarding interpolants in the limit of flat radial basis functions”. *Comput. Math. Appl.* 47 (2004), pp. 37–55.

[46] B. Fornberg and J. Zuev. “The Runge phenomenon and spatially variable shape parameters in RBF interpolation”. *Comput. Math. Appl.* 54 (2007), pp. 379–398.

[47] M. Gasca and T. Sauer. “Polynomial interpolation in several variables”. *Adv. Comput. Math.* 12 (2000), pp. 377–410.

[48] P. Gonnet, R. Pachón, and L. N. Trefethen. “Robust rational interpolation and least-squares”. *Electron. Trans. Numer. Anal.* 38 (2011), pp. 146–167.

[49] L. Grasedyck, W. Hackbusch, and R. Kriemann. “Performance of H-LU preconditioning for sparse matrices”. *Comput. Methods Appl. Math.* 8 (2008), pp. 336–349.

[50] L. Grasedyck, R. Kriemann, and S. Le Borne. “Domain decomposition based $\mathcal{H}$-LU preconditioning”. *Numer. Math.* 112 (2009), pp. 565–600.

[51] L. Grasedyck, R. Kriemann, and S. Le Borne. “Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems”. *Comput. Vis. Sci.* 11 (2008), pp. 273–291.

[52] S. Börm. *H2Lib*. `http://www.h2lib.org/`. 2017. Retrieved February 27, 2021.

[53] W. Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*. Vol. 49. Springer Ser. Comput. Math. 2015.

[54] W. Hackbusch, B. N. Khoromskij, and R. Kriemann. “Hierarchical matrices based on a weak admissibility criterion”. *Computing* 73 (2004), pp. 207–243.

[55] J. H. Halton. “On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals”. *Numer. Math.* 2 (1960), pp. 84–90.

[56] IEEE. “IEEE Standard for Floating-Point Arithmetic”. *IEEE Std 754-2008* (2008), pp. 1–70.

[57] A. Iske. "On the approximation order and numerical stability of local Lagrange interpolation by polyharmonic splines". In: *Modern developments in multivariate approximation*. Vol. 145. Internat. Ser. Numer. Math. 2003, pp. 153–165.

[58] M. Jančič and G. Kosec. "A hybrid RBF-FD and WLS mesh-free strong-form approximation method". *arXiv:2203.02178* (2022).

[59] M. Jančič, J. Slak, and G. Kosec. "p-refined RBF-FD solution of a Poisson problem". In: *2021 6th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE. 2021, pp. 01–06.

[60] M. Jančič, J. Slak, and G. Kosec. "Analysis of high order dimension independent RBF-FD solution of Poisson's equation". *arXiv:1909.01126* (2019).

[61] M. Jančič, J. Slak, and G. Kosec. "Monomial Augmentation Guidelines for RBF-FD from Accuracy Versus Computational Time Perspective". *J. Sci. Comput.* 87 (2021), pp. 1–18.

[62] Mitja Jančič and Gregor Kosec. "Strong form mesh-free hp-adaptive solution of linear elasticity problem". *Engineering with Computers* (2023), pp. 1–21.

[63] A. Javed et al. "Upwind skewed radial basis functions (USRBF) for solution of highly convective problems over meshfree nodes". *Engineering with Computers* 37 (2021), pp. 1081–1097.

[64] G. Kosec. "A local numerical solution of a fluid-flow problem on an irregular domain". *Adv. Eng. Softw.* 120 (2018), pp. 36–44.

[65] R. Kriemann and S. Le Borne. "$\mathcal{H}$-FAINV: hierarchically factored approximate inverse preconditioners". *Comput. Vis. Sci.* 17 (2015), pp. 135–150.

[66] *LAPACK - Linear Algebra PACKage*. `http://www.netlib.org/lapack/`. Retrieved April 19, 2021.

[67] E. Larsson and B. Fornberg. "Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions". *Comput. Math. Appl.* 49 (2005), pp. 103–130.

[68] E. Larsson et al. "Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions". *SIAM J. Sci. Comput.* 35 (2013), A2096–A2119.

[69] A. P. Lawrence, M. E. Nielsen, and B. Fornberg. "Node Subsampling for Multilevel Meshfree Elliptic PDE Solvers". *arXiv:2303.09080* (2023).

[70] D. Lazzaro and L. B. Montefusco. "Radial basis functions for the multivariate interpolation of large scattered data sets". *J. Comput. Appl. Math.* 140 (2002), pp. 521–536.

[71] S. Le Borne. "Factorization, symmetrization, and truncated transformation of radial basis function-GA stabilized Gaussian radial basis functions". *SIAM J. Matrix Anal. Appl.* 40 (2019), pp. 517–541.

[72] S. Le Borne. "Hierarchical matrix preconditioners for the Oseen equations". *Comput. Vis. Sci.* 11 (2008), pp. 147–157.

[73] S. Le Borne and L. Grasedyck. "H-matrix preconditioners in convection-dominated problems". *SIAM J. Matrix Anal. Appl.* 27 (2006), pp. 1172–1183.

[74] S. Le Borne and W. Leinen. "Guidelines for RBF-FD discretization: Numerical experiments on the interplay of a multitude of parameter choices". *J. Sci. Comput.* 95, 8 (2023).

[75] S. Le Borne and W. Leinen. *Smaller stencil preconditioners for polyharmonic spline RBF-FD discretizations*. Preprint available at Research Square [`https://doi.org/10.21203/rs.3.rs-2888756/v1`]. 2023.

[76] E. Lehto. *RBF-GA Differentiation Weights*. `https://www.mathworks.com/matlabcentral/fileexchange/48198-rbf-ga-differentiation-weights`. 2014. MATLAB Central File Exchange. Retrieved July 14, 2020.

[77] N. Li et al. "Multiquadric RBF-FD method for the convection-dominated diffusion problems base on Shishkin nodes". *International Journal of Heat and Mass Transfer* 118 (2018), pp. 734–745.

[78] Y. Liu et al. "A cache-efficient reordering method for unstructured meshes with applications to wall-resolved large-eddy simulations". *J. Comput. Phys.* 480, 112009 (2023).

[79] A. L. Marchildon and D. W. Zingg. "Unisolvency for Polynomial Interpolation in Simplices with Symmetrical Nodal Distributions". *J. Sci. Comput.* 92, 50 (2022).

[80] B. Martin and B. Fornberg. "Using radial basis function-generated finite differences (RBF-FD) to solve heat transfer equilibrium problems in domains with interfaces". *Eng. Anal. Bound. Elem.* 79 (2017), pp. 38–48.

[81] N. H. Mathews, N. Flyer, and S. E. Gibson. "Solving 3D magnetohydrostatics with RBF-FD: Applications to the solar corona". *J. Comput. Phys.* 462, 111214 (2022).

[82] S. Milovanović. "Pricing financial derivatives using radial basis function generated finite differences with polyharmonic splines on smoothly varying node layouts". *arXiv:1808.02365* (2018).

[83] S. Milovanović and L. von Sydow. "Radial Basis Function generated Finite Differences for option pricing problems". *Comput. Math. Appl.* 75 (2018), pp. 1462–1481.

[84] P. K. Mishra. "NodeLab: A MATLAB package for meshfree node-generation and adaptive refinement". *Journal of Open Source Software* 4, 1173 (2019).

[85] P. K. Mishra et al. "A stabilized radial basis-finite difference (RBF-FD) method with hybrid kernels". *Comput. Math. Appl.* 77 (2019), pp. 2354–2368.

[86] P. K. Mishra et al. "Adaptive radial basis function generated finite-difference (RBF-FD) on non-uniform nodes using p-refinement". *arXiv:2004.06319* (2020).

[87] P. K. Mishra et al. "An improved radial basis-pseudospectral method with hybrid Gaussian-cubic kernels". *Eng. Anal. Bound. Elem.* 80 (2017), pp. 162–171.

[88] V. Mohammadi, M. Dehghan, and S. De Marchi. "Numerical simulation of a prostate tumor growth model by the RBF-FD scheme and a semi-implicit time discretization". *J. Comput. Appl. Math.* 388, 113314 (2021).

[89] D. M. Mount and S. Arya. *ANN: A Library for Approximate Nearest Neighbor Searching*. `http://www.cs.umd.edu/~mount/ANN/`. 2010. Retrieved June 25, 2018.

[90] Y. Notay. "Aggregation-based algebraic multigrid for convection-diffusion equations". *SIAM J. Sci. Comput.* 34 (2012), A2288–A2316.

[91] D. T. Oanh, O. Davydov, and H. X. Phu. "Adaptive RBF-FD method for elliptic problems with point singularities in 2D". *Appl. Math. Comput.* 313 (2017), pp. 474–497.

[92] D. T. Oanh and N. M. Tuong. "An approach to adaptive refinement for the RBF-FD method for 2D elliptic equations". *Appl. Numer. Math.* 178 (2022), pp. 123–154.

[93] Ö. Oruç. "A local hybrid kernel meshless method for numerical solutions of two-dimensional fractional cable equation in neuronal dynamics". *Numer. Methods Partial Differential Equations* 36 (2020), pp. 1699–1717.

[94] A. Radhakrishnan et al. "A non-nested multilevel method for meshless solution of the Poisson equation in heat transfer and fluid flow". *arXiv:2104.13758* (2021).

[95] *rand*. `https://en.cppreference.com/w/c/numeric/random/rand`. Retrieved June 11, 2019.

[96] R. J. Renka. "Multivariate interpolation of large sets of scattered data". *ACM Trans. Math. Software* 14 (1988), pp. 139–148.

[97] C. M. C. Roque et al. "A local radial basis functions-Finite differences technique for the analysis of composite plates". *Eng. Anal. Bound. Elem.* 35 (2011), pp. 363–374.

[98] M. Rossini. "Interpolating functions with gradient discontinuities via variably scaled kernels". *Dolomites Res. Notes Approx.* 11 (2018), pp. 3–14.

[99] M. Rot and A. Rashkovska. "Meshless method stencil evaluation with machine learning". *arXiv:2204.12940* (2022).

[100] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[101] K. van der Sande and B. Fornberg. "Fast variable density 3-D node generation". *SIAM J. Sci. Comput.* 43 (2021), A242–A257.

[102] L. G. C. Santos et al. "Comparing RBF-FD approximations based on stabilized Gaussians and on polyharmonic splines with polynomials". *Internat. J. Numer. Methods Engrg.* 115 (2018), pp. 462–500.

[103] Y. V. S. S. Sanyasiraju and C. Satyanarayana. "Upwind strategies for local RBF scheme to solve convection dominated problems". *Eng. Anal. Bound. Elem.* 48 (2014), pp. 1–13.

[104] R. Schaback. "Multivariate Interpolation by Polynomials and Radial Basis Functions". *Constr. Approx.* 21 (2005), pp. 293–317.

[105] B. Seibold. "Minimal positive stencils in meshfree finite difference methods for the Poisson equation". *Comput. Methods Appl. Mech. Engrg.* 198 (2008), pp. 592–601.

[106] V. Shankar. "The overlapped radial basis function-finite difference (RBF-FD) method: A generalization of RBF-FD". *J. Comput. Phys.* 342 (2017), pp. 211–228.

[107] V. Shankar and A. L. Fogelson. "Hyperviscosity-based stabilization for radial basis function-finite difference (RBF-FD) discretizations of advection-diffusion equations". *J. Comput. Phys.* 372 (2018), pp. 616–639.

[108] V. Shankar, R. M. Kirby, and A. L. Fogelson. "Robust node generation for mesh-free discretizations on irregular domains and surfaces". *SIAM J. Sci. Comput.* 40 (2018), A2584–A2608.

[109] V. Shankar, G. B. Wright, and A. L. Fogelson. "An efficient high-order meshless method for advection-diffusion equations on time-varying irregular domains". *J. Comput. Phys.* 445, 110633 (2021).

[110] C. Shu et al. "An upwind local RBF-DQ method for simulation of inviscid compressible flows". *Comput. Methods Appl. Mech. Engrg.* 194 (2005), pp. 2001–2017.

[111] J. Slak. "Adaptive RBF-FD method". PhD thesis. University of Ljubljana, 2020.

[112] J. Slak and G. Kosec. "Medusa: A C++ Library for Solving PDEs Using Strong Form Mesh-Free Methods". *ACM Trans. Math. Software* 47 (2021), pp. 1–25.

[113] J. Slak and G. Kosec. "On Generation of Node Distributions for Meshless PDE Discretizations". *SIAM J. Sci. Comput.* 41 (2019), A3202–A3229.

[114] J. Slak and G. Kosec. “Refined Meshless Local Strong Form solution of Cauchy-Navier equation on an irregular domain”. *Eng. Anal. Bound. Elem.* 100 (2018), pp. 3–13.

[115] J. Slak and G. Kosec. *Standalone implementation of the proposed node placing algorithm.* `http://e6.ijs.si/medusa/static/PNP.zip`. Retrieved January 11, 2022.

[116] W. F. Spotz and G. F. Carey. “A high-order compact formulation for the 3D Poisson equation”. *Numer. Methods Partial Differential Equations* 12 (1996), pp. 235–243.

[117] L. Su. “A radial basis function (RBF)-finite difference (FD) method for the backward heat conduction problem”. *Appl. Math. Comput.* 354 (2019), pp. 232–247.

[118] M. Tillenius et al. “A scalable RBF-FD method for atmospheric flow”. *J. Comput. Phys.* 298 (2015), pp. 406–422.

[119] A. I. Tolstykh. “On using RBF-based differencing formulas for unstructured and mixed structured-unstructured grid calculations”. In: *Proceedings of the 16th IMACS world congress.* Vol. 228. 2000, pp. 4606–4624.

[120] A. I. Tolstykh and D. A. Shirobokov. “On using radial basis functions in a “finite difference mode” with applications to elasticity problems”. *Comput. Mech.* 33 (2003), pp. 68–79.

[121] I. Tominec, E. Larsson, and A. Heryudono. “A least squares radial basis function finite difference method with improved stability properties”. *SIAM J. Sci. Comput.* 43 (2021), A1441–A1471.

[122] B. Tóth and A. Düster. “h-Adaptive radial basis function finite difference method for linear elasticity problems”. *Comput. Mech.* 71 (2023), pp. 433–452.

[123] J. G. van der Corput. “Verteilungsfunktionen. I. Mitt.” German. *Proc. Akad. Wet. Amsterdam* 38 (1935), pp. 813–821.

[124] C. B. Vreugdenhil. *Numerical Methods for Shallow-Water Flow.* Vol. 13. Water Science and Technology Library. Springer Science & Business Media, 2013.

[125] A. J. Wathen. “Preconditioning”. *Acta Numer.* 24 (2015), pp. 329–376.

[126] H. Wendland. “Computational Aspects of Radial Basis Function Approximation”. In: *Topics in Multivariate Approximation and Interpolation.* Vol. 12. Studies in Computational Mathematics. Elsevier, 2006, pp. 231–256.

[127] H. Wendland. *Scattered Data Approximation.* Cambridge Monogr. Appl. Comput. Math. 2005.

[128] G. B. Wright and B. Fornberg. “Scattered node compact finite difference-type formulas generated from radial basis functions”. *J. Comput. Phys.* 212 (2006), pp. 99–123.

[129] G. B. Wright and B. Fornberg. "Stable computations with flat radial basis functions using vector-valued rational approximations". *J. Comput. Phys.* 331 (2017), pp. 137–156.

[130] G. B. Wright, A. Jones, and V. Shankar. "MGM: A meshfree geometric multilevel method for systems arising from elliptic equations on point cloud surfaces". *SIAM J. Sci. Comput.* 45 (2023), A312–A337.

[131] P. N. Yianilos. "Data structures and algorithms for nearest neighbor search in general metric spaces". In: *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*. SODA '93. 1993, pp. 311–321.

[132] R. Zamolo, D. Miotti, and E. Nobile. "Accurate Stabilization Techniques for RBF-FD Meshless Discretizations with Neumann Boundary Conditions". *arXiv:2207.06725* (2022).

[133] R. Zamolo, E. Nobile, and B. Šarler. "Novel multilevel techniques for convergence acceleration in the solution of systems of equations arising from RBF-FD meshless discretizations". *J. Comput. Phys.* 392 (2019), pp. 311–334.

[134] R. Zamolo and L. Parussini. "Geometric uncertainty propagation in laminar flows solved by RBF-FD meshless technique". In: *Journal of Physics: Conference Series*. Vol. 1599. IOP Publishing. 2020.

[135] R. Zamolo, L. Parussini, and E. Nobile. "Propagation of geometric uncertainties in heat transfer problems solved by RBF-FD meshless method". In: *Journal of Physics: Conference Series*. Vol. 1868. IOP Publishing. 2021.

[136] G. Zhang. "Smoothing splines using compactly supported, positive definite, radial basis functions". *Comput. Statist.* 27 (2012), pp. 573–584.

[137] F. Zhao et al. "The characteristic RBF-FD method for the convection-diffusion-reaction equation on implicit surfaces". *Numerical Heat Transfer, Part A: Applications* 75 (2019), pp. 548–559.