Two algorithms for fast 2D node generation: Application to RBF meshless discretization of diffusion problems and image halftoning

Riccardo Zamolo *, Enrico Nobile

Department of Engineering & Architecture, University of Trieste, via Alfonso Valerio 10, 34127 Trieste, Italy

ARTICLE INFO

Accepted 15 March 2018

Keywords:
Meshless methods
Local RBF
Node/point generation algorithms
Quadtree
Dithering
Stippling

ABSTRACT

Mesh generation techniques for traditional mesh based numerical approaches such as FEM and FVM have now reached a good degree of maturity. There is no such an acknowledged background when dealing with node generation techniques for meshless numerical approaches, despite their theoretical simplicity and efficiency; furthermore node generation can be put in connection with some well-known image approximation techniques. Two node generation algorithms are here proposed and employed in the numerical solution of 2D steady state diffusion problems by means of a local Radial Basis Function (RBF) meshless method. Finally, such algorithms are also tested for grayscale image approximation through stippling.

1. Introduction

The introduction and development of mesh based numerical techniques for the discretization of PDEs have led to an analogous growth and improvement of mesh generation techniques [1]. In particular, unstructured meshing is crucial when dealing with engineering relevant geometries with generic shapes; therefore there is a wide and well established knowledge in the field of unstructured mesh generation [2].

Emerging numerical techniques such as meshless (or mesh-free) approaches for the solution of PDEs [3–5] only require a proper node distribution and no additional geometrical data to define a spatial discretization of the problem; obviously, this appealing advantage over mesh-based approaches relies upon the availability of node generation techniques which have to be simple, fast and robust.

Up to now, only a few node generation algorithms have been proposed for the specific use with meshless methods [6–8]; nonetheless such approaches confirmed that node generation can be easier and faster than traditional mesh generation. Recently, another advancing-front type algorithm has been proposed for 2D node generation [9], which showed the possibility to deal with complex boundaries and strong node density variations, still maintaining computational speed and algorithmic simplicity.

In this work we propose two different algorithms for the fast generation of 2D node distributions which are suitable for RBF meshless discretization of PDEs: an initial node distribution, which satisfies the prescribed nodal spacing, is generated and then iteratively refined in order to obtain a locally isotropic node distribution; both algorithms are characterized by simplicity and high node generation rates (≈65 000 nodes per second on a single core).

* Corresponding author.
E-mail addresses: riccardo.zamolo@phd.units.it (R. Zamolo), nobile@units.it (E. Nobile).

Several test cases are carried out using the node distributions generated by the proposed algorithms in order to solve 2D steady state heat conduction problems (diffusion problems) through a local RBF collocation meshless method; the results are also compared to Finite Element Method (FEM) and Finite Difference (FD) approaches, showing that the coupling of our node generation algorithms with the above-mentioned meshless method can be highly effective in the practical simulation of heat conduction problems (but not only) over complex shaped domains.

Another application of node generation algorithms is stippling, a technique used for the approximation of grayscale images through the 2D placing of scattered points; some visual test cases are also presented for this problem in order to show how the proposed algorithms can also be used for an efficient and accurate halftoning approximation of grayscale images.

2. 2D Node generation algorithms

2.1. Inputs of the problem

The node generation problem here considered assumes a prescribed spacing function $s(\mathbf{x})$ and the shape of the domain Ω as input data to generate an isotropic distribution of nodes $X(s)$ within Ω. The prescribed spacing function $s(\mathbf{x})$ defines the local linear spacing between the nodes, and in 2D it is formally defined by:

$$\frac{1}{s^2(\mathbf{x})} = \lim_{\substack{A\to\mathbf{x} \\ k\to+\infty}} \frac{\#nodes(X(s/k), A)}{k^2\mu(A)} \tag{1}$$

where $k \in \mathbb{R}^+$, A is a portion of Ω whose measure (area) is given by $\mu(A)$ and $\#nodes(X, A)$ gives the number of nodes of distribution X lying inside A; in the limit (1) A must satisfy $\mu(A) \geq ck^{-\alpha}$ for some constants $\alpha < 2$ and $c > 0$.

2.2. Notes on iterative node repel approach

Both proposed algorithms are characterized by two phases: an initial node placing phase and an iterative refinement phase; the initial phase creates a distribution whose node spacing matches the prescribed spacing function $s(\mathbf{x})$ except for some high spatial frequency error that is smoothed out in the refinement phase which improves the local quality and isotropy of the distribution, while a fixed boundary node distribution, obeying $s(\mathbf{x})$, is considered; in the refinement phase, based upon a node repel algorithm, nodes move according to the mutual radial repulsion forces of the 9 nearest neighboring nodes:

$$\Delta\mathbf{x}_i = \omega s(\mathbf{x}_i)\sum_{NB=1}^{9} \mathbf{e}_{ji}\left[4\left(\frac{r_{ij}}{\bar{s}_{ij}}\right)^2 + 1\right]^{-2} \tag{2}$$

where $\Delta\mathbf{x}_i$ is the displacement of node $\mathbf{x}_i$, $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ is the distance between a couple of nodes i and j, $j = i_{NB}$ is the index of the NBth nearest neighbor of node i, $\mathbf{e}_{ji} = (\mathbf{x}_i - \mathbf{x}_j)/r_{ij}$ is the unit vector from node j to node i, $\bar{s}_{ij} = [s(\mathbf{x}_i) + s(\mathbf{x}_j)]/2$ is the mean spacing between the two nodes and $\omega \in [0.5, 1]$ is a relaxation parameter.

The use of 9 neighbors in Eq. (2) allows a better behavior of the refinement process during the initial iterations when the local node anisotropy can be very high. The choice of the actual nearest neighboring nodes is performed after each single repel step since this operation has positive effect on the node distribution quality and has a negligible cost (only few squared distances have to be evaluated).

Eq. (2) is asymptotically correct in the following sense: if an isotropic distribution of nodes $X(s)$ satisfies $r_{ij} = \bar{s}_{ij}$ for each node i and for each of its 9 neighbors $j = i_{NB}$, then $\Delta\mathbf{x}_i/s(\mathbf{x}_i)$ depends only on the local node anisotropy, which is by definition "small" or zero, and therefore node displacements will be "small" or zero; the contrary does not hold: if the refinement process converges, the final stable node configuration is not guaranteed to satisfy $r_{ij} = \bar{s}_{ij}$ (consider a uniform cartesian node distribution and a constant spacing function), but hopefully this last condition will be "sufficiently" satisfied.

A two level strategy is employed to improve the iterative refinement: we found that an effective choice is to alternate 1 iteration with a relaxation factor $\omega = 0.9$ followed by 7 iterations with $\omega = 0.5$, therefore the number of iterations here presented will be multiple of 8. In fact, ω acts as a multiplier in Eq. (2), setting the amount of displacement for each node: the choice $\omega = 0.9$ (which would cause the divergence of the process if iterated indefinitely) for just 1 iteration can partially avoid the stagnation of the refinement process when locally equilibrated configurations arise, since large displacements are allowed for that single iteration. We note that this problem could also be avoided introducing some random displacement $\Delta\mathbf{x}_i = \omega_R s(\mathbf{x}_i) U\mathbf{e}_U$, with U a uniform random variable in $[-1, 1]$, $\mathbf{e}_U$ a uniform random unit vector and $\omega_R \in (0, 0.5]$ the corresponding displacement multiplier.

The initial node placing phase is required because the node repel iterative refinement algorithm efficiently smooths out only the high spatial frequency component of the error between the spacing of the node distribution and the prescribed spacing s, while the low spatial frequency component would require a very high number of iterations to be significantly reduced.

This property is highlighted by Fig. 1, where two initial 1-D distributions with high and low frequency error are subjected to refinement iterations; in the first case the deviation $\sigma[\Delta x]$ between the spacing Δx within nodes and the prescribed spacing $s = cost$ rapidly decreases to 0 within a small number of repel iterations, while in the second case the deviation stagnates for a large number of repel iterations.

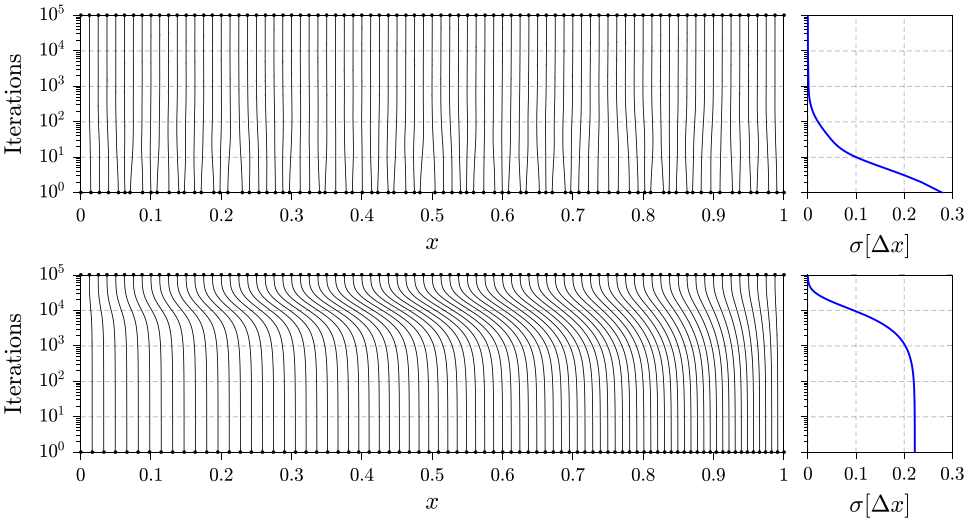

Fig. 1. Refinement algorithm evolution for 1-D initial distributions with high (top) and low (bottom) frequency error in the case of a prescribed spacing $s = cost$ and $N = 80$ nodes.

2.3. The Overlapping Crossed Columns (OXC) algorithm

Starting from the smallest box (aligned with $x-y$ directions) bounding domain Ω, this algorithm proceeds by filling each vertical strip in which the box is partitioned; these operations are then repeated reversing the spatial directions, therefore proceeding on horizontal strips.

The pseudocode for this procedure is reported in Algorithm 1, where:

- $\mathcal{S}(BoundingBox, s, dir)$ are the vertical ($dir = d_x$) or horizontal ($dir = d_y$) strips in which *BoundingBox* is partitioned; each of these strips has width w along direction *dir* such that $w = \max s$ within each strip;
- GenerateNodesOXC($\cdot$, *Strip*, s) fills the whole *Strip* using spacing s;
- LinearDistribution($\cdot$, *Strip*, s) fills *Strip* as long as $w/2 < s \leq w$, where w is the width of *Strip*;
- $\mathcal{H}(Strip)$ are the two vertical substrips of the unfilled portion of *Strip*;
- ParentStrip(*Strip*) is the unfilled portion of the parent strip of *Strip*.

Algorithm 1 Overlapping Crossed Columns (OXC) 2D node generator

Input: domain Ω, prescribed spacing function s
Output: *NodeDistribution*

```
NodeDistribution ← empty node distribution
BoundingBox ← smallest box bounding domain Ω
for each dir = d_x, d_y do
    for each Strip ∈ S(BoundingBox,√2 s,dir) do
        GenerateNodesOXC(NodeDistribution,Strip,√2 s)
function GenerateNodesOXC(Nodes,Strip,s)
    StripState ← LinearDistribution(Nodes,Strip,s)
    switch StripState do
        case s decreasing
            for each SubStrip ∈ H(Strip) do
                GenerateNodesOXC(Nodes,SubStrip,s)
        case s increasing
            ParentStrip ← ParentStrip(Strip)
            GenerateNodesOXC(Nodes,ParentStrip,s)
end function
```

We point out that the $\sqrt{2}$ factor for spacing s in Algorithm 1 is due to the double call to GenerateNodesOXC in order to ensure the correct final spacing.

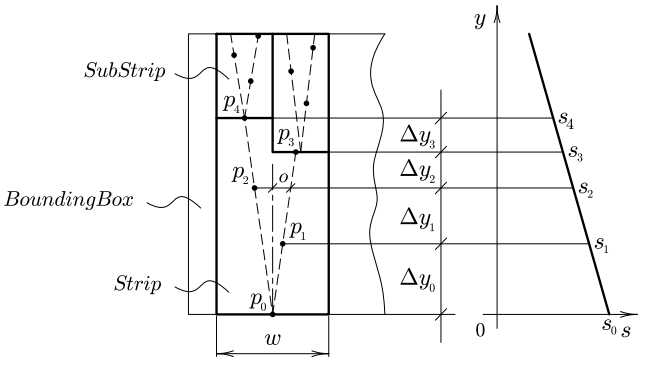

Fig. 2. Node placing with OXC algorithm.

A graphical representation of the procedure is reported in Fig. 2 in the case of a linear spacing s decreasing with y; LinearDistribution(*Nodes*, *Strip*, s) fills *Strip* placing nodes with an alternating horizontal offset $o = \pm(w - s)/2$ from the vertical midline, while the vertical offset is $\Delta y = s^2/w$ which guarantees a correct mean spacing within *Strip*.

Function LinearDistribution, which is the core of the OXC algorithm, consists of roughly 50 lines of C code, while the whole OXC algorithm has been implemented in roughly 150 lines of C code.

To the best of authors' knowledge, the principles of OXC algorithm here presented have never been previously considered in literature for node/point generation processes, and are here introduced for the first time.

2.4. The Modified Quadtree (MQT) algorithm

The Quadtree algorithm [10] is a widely used space partitioning technique; mesh generation using such algorithm is also widely known [11], while an example of quadtree node generator for meshless discretizations can be found in [8]. In order to correctly account for space varying node spacing, we propose a modification of the original algorithm as reported in Algorithm 2, where:

- InitializeNumerosity(s, *BoundingBox*) initializes a numerosity function that gives the prescribed number of nodes, evaluated from prescribed spacing s, to be contained in each square sub-box of *BoundingBox*;
- GenerateNodesMQT($\cdot$, *Box*, M) fills the whole square *Box* using numerosity function M;
- InsertNodes($\cdot$, *Box*, i) fills *Box* with an integer number $i \leq 3$ of nodes;
- DitheringMQT(M, *err*, *Box*) is the Quadtree version of the dithering algorithm: the quantization error *err* of *Box* is spread onto the unvisited neighboring boxes, correcting the numerosity function M;
- $\mathcal{Q}$(*Box*) are the four square sub-boxes of *Box*;

Algorithm 2 Modified Quadtree (MQT) 2D node generator

Input: domain Ω, prescribed spacing function s
Output: *NodeDistribution*

```
1:  NodeDistribution ← empty node distribution
2:  BoundingBox ← smallest square bounding domain Ω
3:  M_s ← InitializeNumerosity(s,BoundingBox)
4:  GenerateNodesMQT(NodeDistribution,BoundingBox,M_s)
5:  function GenerateNodesMQT(Nodes,Box,M)
6:      m ← M(Box)
7:      if ⌊m⌉ ≤ 3 then
8:          InsertNodes(Nodes,Box,⌊m⌉)
9:          QuantizationError ← ⌊m⌉ − m
10:         DitheringMQT(M,QuantizationError,Box)
11:     else
12:         for each SubBox ∈ Q(Box) do
13:             GenerateNodesMQT(Nodes,SubBox,M)
14: end function
```

The main modification of the traditional Quadtree approach is the introduction of a dithering correction of the numerosity function as the node placing proceeds; dithering is a widely known image processing algorithm which diffuses the

Table 1
Features of OXC and MQT algorithms.

Algorithm	Complexity	Memory	Robustness
OXC	$\mathcal{O}(N)$	$\mathcal{O}(N)$	Low
MQT	$\mathcal{O}(N \log s_{min}^{-1})$	$\mathcal{O}(s_{min}^{-2})$	High

quantization error of a pixel onto its neighboring pixels in order to maintain a negligible mean error across the image. We chose the well known Floyd–Steinberg dithering algorithm [12], whose weights stencil is the following:

$$\begin{bmatrix} - & \circ & 7/16 \\ 3/16 & 5/16 & 1/16 \end{bmatrix} \tag{3}$$

where $\circ$ is the pixel being processed and the algorithm proceeds by lexicographic order. This algorithm is implemented in order to diffuse the quantization error between the integer number of nodes that are placed in a box and the corresponding non-integer prescribed numerosity through the Quadtree data structure; this task is performed by DitheringMQT, which in our implementation consists of roughly 30 lines of C code, while the whole MQT algorithm has been implemented in roughly 200 lines of C code.

2.5. Features of OXC and MQT algorithms

Table 1 summarizes three main features of both proposed algorithms, namely, algorithmic complexity, memory consumption and robustness against strong node density variations.

The OXC algorithm is the simplest and has no special needs for the data structure: only a linear array whose length is dynamically adjusted during the computation is needed to store the coordinates of the nodes; both complexity and memory consumption are linear in the number of nodes N. These favorable characteristics are counterbalanced by the low robustness against steep/discontinuous spacing functions: the OXC algorithm, by its nature within current implementation, can produce wrong node distributions for strong node density variations which are not aligned with the cartesian axes.

On the contrary, the MQT algorithm has a slightly less straightforward implementation since it needs a Quadtree hierarchical data structure in order to diffuse quantization errors on different hierarchical levels. This non-trivial data structure, which requires $\mathcal{O}(s_{min}^{-2})$ memory where s_{min} is the minimum value of the spacing function, implies a complexity that also depends upon s_{min}; these properties are justified by the high robustness against strong or even discontinuous spacing functions.

3. Local RBF collocation meshless discretization

3.1. Problem definition

In order to test and compare the proposed node generation algorithms when coupled with a meshless approach, a RBF-based meshless method is employed to discretize the 2D Poisson equation:

$$\nabla^2 T = q \tag{4}$$

defined on domain Ω, and subjected to Dirichlet boundary conditions $T = \bar{T}$ on Γ_D and Neumann boundary conditions $\partial T/\partial \mathbf{n} = \mathbf{n} \cdot \nabla T = \bar{f}$ on Γ_N, where $\Gamma_D \cup \Gamma_N = \partial\Omega$ is the domain boundary and $\mathbf{n}$ is the exterior normal to the boundary. Eq. (4) is representative of steady state diffusion problems, such as heat conduction in a homogeneous media with internal heat generation.

3.2. Numerical method

We briefly present the local Radial Basis Function collocation meshless method employed in [13,14] for the numerical solution of thermal problems.

RBF interpolation.

The unknown field T around $\mathbf{x}$ is approximated through the following expansion:

$$T(\mathbf{x}) = \sum_{j \in J_n(\mathbf{x})} a_j \varphi(\|\mathbf{x} - \mathbf{x}_j\|) + \mathbf{b} \cdot \mathbf{x} + c \tag{5}$$

where $J_n(\mathbf{x})$ represents the indices j of the n nodes $\mathbf{x}_j$ closest to $\mathbf{x}$. $T(\mathbf{x})$ is therefore a linear combination of n radial functions φ centered at the n local nodes $\mathbf{x}_j$ plus a linear polynomial in $\mathbf{x}$.

Hardy's Multiquadric (MQ) [15] has been chosen as RBF:

$$\varphi(r) = \sqrt{1 + (\varepsilon r)^2} \tag{6}$$

where the shape factor is rescaled as $\varepsilon = s_{max}\bar{\varepsilon}/d_n$: $\bar{\varepsilon}$ is the rescaled shape factor, s_{max} is the maximum prescribed spacing on Ω and d_n is the local subdomain size, defined as the root mean square distance of the n local nodes from their midpoint.

The coefficients a_j, $\mathbf{b}$ and c can be computed writing Eq. (5) for the $m \leq n$ local nodes which do not lie on the Neumann boundary Γ_N:

$$T(\mathbf{x}_i) = T_i \tag{7}$$

where T_i is the unknown field in $\mathbf{x}_i$. The following additional relations are needed because of the linear polynomial in (5):

$$\sum_{j=1}^{n} a_j = 0 \quad , \quad \sum_{j=1}^{n} x_j a_j = 0 \quad , \quad \sum_{j=1}^{n} y_j a_j = 0. \tag{8}$$

If any of the local nodes lies on the Neumann boundary, for each of these $n-m$ Neumann nodes the corresponding boundary condition must be satisfied:

$$\mathbf{n} \cdot \nabla T(\mathbf{x}_i) = \bar{f}_i \tag{9}$$

where $\nabla T(\cdot)$ is given by the gradient of Eq. (5).

Collecting the n coefficients a_i, the m unknown values T_i and the $n-m$ Neumann boundary contributions $\bar{f}_i$ in column vectors $\mathbf{a}$, $\mathbf{T}$ and $\bar{\mathbf{f}}$, respectively, the interpolation system, in compact form, is the following:

$$\mathbf{G} \begin{Bmatrix} \mathbf{a} \\ \mathbf{b} \\ c \end{Bmatrix} = \begin{Bmatrix} \mathbf{T} \\ \mathbf{0} \\ \bar{\mathbf{f}} \end{Bmatrix} \tag{10}$$

In the perspective of developing simple and light approaches for the solution of practical problems for which a 2nd order accuracy can be enough, we chose a small number of local nodes n varying from 6 to 8; larger stencil sizes are not considered in this work, in spite of the benefits reported in [16,17] where stencil sizes varied in the range $10 < n < 100$, obtaining up to 9th order accuracy.

Finally, we note that the asymptotic number of edges per node in a triangular mesh with recursive partitioning is exactly 6 [18], corresponding to the choice $n = 7$; an analogous behavior is observed with the node refinement process previously described: each node tends to be surrounded by 6 nearest neighbors, which is an obvious fact since it is the configuration which maximizes the node density.

Collocation.

Eq. (4) with RBF approximation (5) becomes:

$$\nabla^2 T(\mathbf{x}) = \sum_{j \in J_n(\mathbf{x})} a_j \nabla^2 \varphi(\|\mathbf{x} - \mathbf{x}_j\|) = q(\mathbf{x}). \tag{11}$$

Writing Eq. (11) for a generic node $\mathbf{x}_k$ gives:

$$\mathbf{a}^T \mathbf{L}(\mathbf{x}_k) = q(\mathbf{x}_k) \tag{12}$$

where $\mathbf{L}(\mathbf{x}_k)$ is the column vector of the Laplacian of φ evaluated in $\mathbf{x}_k$ for each of the n neighbors $\mathbf{x}_j$. Recalling $\mathbf{a}^T$ from Eq. (10) we obtain:

$$\left\{ \mathbf{T}^T \ \mathbf{0} \ \bar{\mathbf{f}}^T \right\} [\mathbf{G}^T]_a^{-1} \mathbf{L}(\mathbf{x}_k) = q(\mathbf{x}_k) \tag{13}$$

where $[\mathbf{G}^T]_a^{-1}$ is the left $(n+3) \times n$ submatrix of $[\mathbf{G}^T]^{-1}$. Eq. (13) made valid for each node $\mathbf{x}_k$ which does not lie on the boundary gives the following linear system:

$$\mathbf{A}_{II} \mathbf{T}_I = \mathbf{q} - \mathbf{A}_{ID} \mathbf{T}_D - \bar{\mathbf{f}}_{Ne} \tag{14}$$

where the subscripts I, D and Ne refer to *Internal*, *Dirichlet* and *Neumann* contributions, respectively; system (14) has been solved through direct LU decomposition of $\mathbf{A}_{II}$ since it is unsymmetric.

Error norms.

The comparison between the computed solution T_{comp} and the corresponding analytical solution T_{an} has been done computing the normalized root mean square norm of the error (NRMSE):

$$\text{NRMSE} = \sqrt{\frac{1}{\mu(\Omega)} \int_{\Omega} \left(\frac{T_{comp} - T_{an}}{T_{max} - T_{min}} \right)^2 d\Omega} \tag{15}$$

where $\mu(\Omega)$ is the measure (area) of Ω; the 2D integral has been approximated by numerical quadrature using the squared prescribed spacing function s^2 as quadrature weight (s^2 is an approximation of the area around each node).

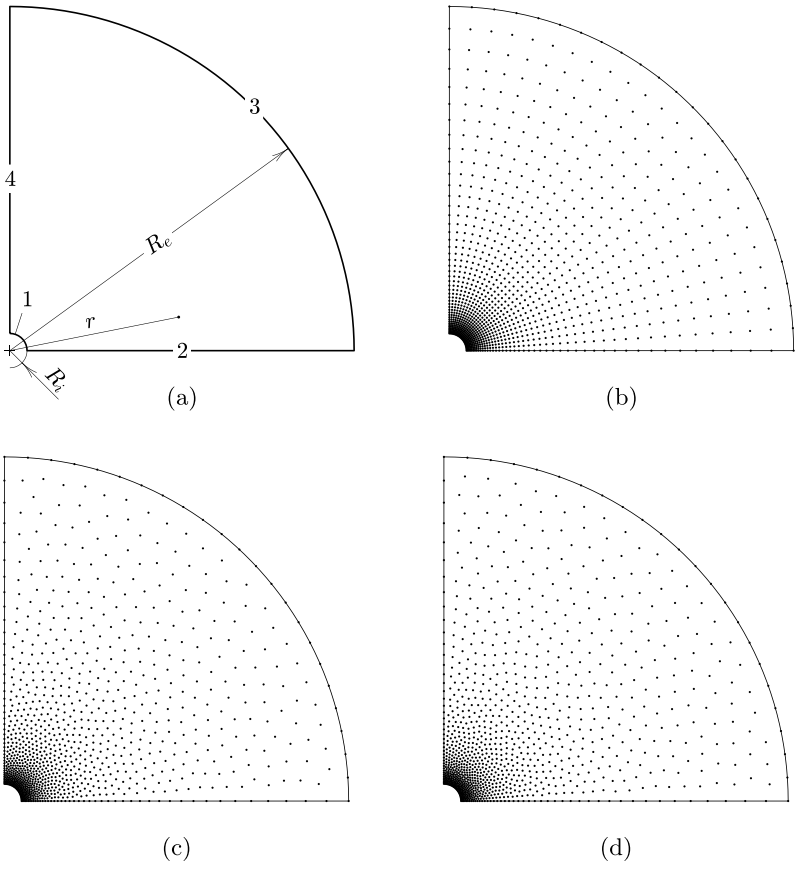

Fig. 3. Test case A: geometry (a) and node distributions: specific/regular (b), OXC (c) and MQT (d), $N \approx 1000$ nodes.

Finite Element results have been obtained using COMSOL Multiphysics® FEA software using triangular meshes with linear elements (3 nodes); the number of mesh nodes is kept as close as possible to the number of unknowns of the corresponding meshless solution, as well as the element sizing which is chosen to be as close as possible to the prescribed spacing function $s(\mathbf{x})$.

The computer times for FEM solution (FE discretization and system solution) are in the same order of magnitude of the ones for meshless solution (local RBF discretization and system solution) here presented; this is mainly due to the choice of linear elements in FEM for which the stencil has approximately 7 nodes using high quality triangular meshes, which is very close to the stencil size $n = 6 \div 8$ for the present meshless discretization.

4. Results

4.1. Test case A: 1/4 of a circular annulus

The geometry of the problem is reported in Fig. 3a where the center of the circular sector is also the origin for the radial r coordinate; the ratio R_i/R_e is chosen to be 0.05. The chosen analytical solution is the following:

$$T_{an} = \log(r/R_e) \tag{16}$$

which is harmonic: $q = \nabla^2 T_{an} = 0$, while the prescribed spacing function is chosen to be $s \propto r$.

A specific and regular geometry dependent node distribution is reported in Fig. 3b, while Fig. 3c and 3d show the node distributions generated by the proposed approaches, OXC and MQT respectively.

The following boundary conditions have been considered:

- Case A1: Dirichlet BC on the whole boundary;
- Case A2: Neumann BC on sides 2 and 4, Dirichlet BC elsewhere.

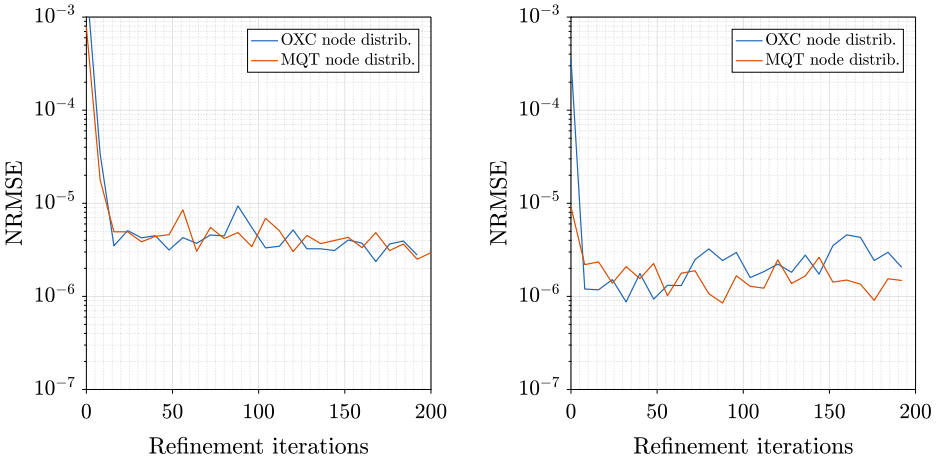

Fig. 4. Test case A1: errors vs number of refinement iterations for $n = 6$ (left) and $n = 8$ (right) neighboring nodes, $N \approx 10\,000$ nodes and $\bar{\varepsilon} = 1$.

First of all we investigated the effect of the number of refinement iterations on the error for $n = 6$ and $n = 8$, Fig. 4, for case A1 with $N \approx 10\,000$ nodes and $\bar{\varepsilon} = 1$. For both cases and both generation algorithms it can be seen that the use of the refinement algorithm is extremely effective in the error reduction within the first $15 \div 25$ iterations, while beyond that point the error shows no significant improvement; for this reason we chose a fixed number of 24 refinement iterations for every other result within this test case.

When dealing with Multiquadrics RBF, it is important to carry out a preliminary sensitivity analysis in order to choose the appropriate rescaled shape factor $\bar{\varepsilon}$; such analysis is reported in Fig. 5 for both test cases A1 and A2, for each number of local nodes $n = 6, 7, 8$ and $N \approx 10\,000$ nodes; FEM results are also displayed as reference.

We first point out that rescaled shape factors $\bar{\varepsilon}$ below 1 have little influence on the error, while above 1 the error can grow by a factor of 10 or 100; for this reason we chose $\bar{\varepsilon} = 1$ for the following results presented in this test case. Another interesting fact that can be observed is that errors obtained with the regular node distribution are always larger than the errors from the refined distributions (MQT and OXC); this can be due to the fact that the regular node distribution here employed has optimal subdomain node arrangements with $n = 9$ nodes, while with $n < 9$ the refined distributions are more effective.

The $\bar{\varepsilon}$ dependency curves also show the influence of boundary conditions: for each case $n = 6, 7, 8$ the error grows by roughly a factor of $5 \div 10$ passing from A1 (Dirichlet BC) to A2 (Mixed BC), confirming the problem of sensitivity to boundary conditions in local RBF approaches. However, an appropriate choice of the rescaled shape factor, for example $\bar{\varepsilon} = 1$, can lead to small errors than FEM in each of the presented cases if refined distributions are employed.

Fig. 6 shows the convergence curves (*i.e.* *NRMSE* vs total number of nodes N) for both test cases A1 and A2, for each number of local nodes $n = 6, 7, 8$ and $\bar{\varepsilon} = 1$. Again, the regular node distribution produced greater errors than refined distributions, although an almost perfect 2nd order accuracy behavior in space (*i.e.* $NRMSE \sim N^{-p/2}$ for $p = 2$ in 2D) for each of the considered cases; the refined distributions reveal also 2nd order accuracy for each case, despite an irregular behavior.

Besides a common 2nd order accuracy, an important conclusion that can be desumed from the analysis of convergence curves is that the employment of refined distributions (MQT and OXC) always produces better results, in terms of error, compared to regular distributions; finally, MQT and OXC node distributions show similar convergence curves in all cases.

The following points summarize the results for this test case:

- A small number of node refinement iterations (≈ 24) is enough to reduce the final error by some orders of magnitude, for both OXC and MQT node distributions;
- Refined OXC and MQT node distributions always allow smaller errors than the regular node distributions for the small stencil sizes here employed ($n < 9$);
- Nearly 2nd order accuracy in each case;
- Significant influence of Neumann BC;
- Small shape factors $\bar{\varepsilon}$ are preferable, within the limits of numerical stability (well-known fact).

4.2. Test case B: L shaped domain

The geometry of the problem is reported in Fig. 7a, while the chosen analytical solution is the following:

$$T_{an} = \left(\frac{r}{L}\right)^{\beta} \sin(\beta\vartheta)\,, \qquad \beta = 2/3 \tag{17}$$

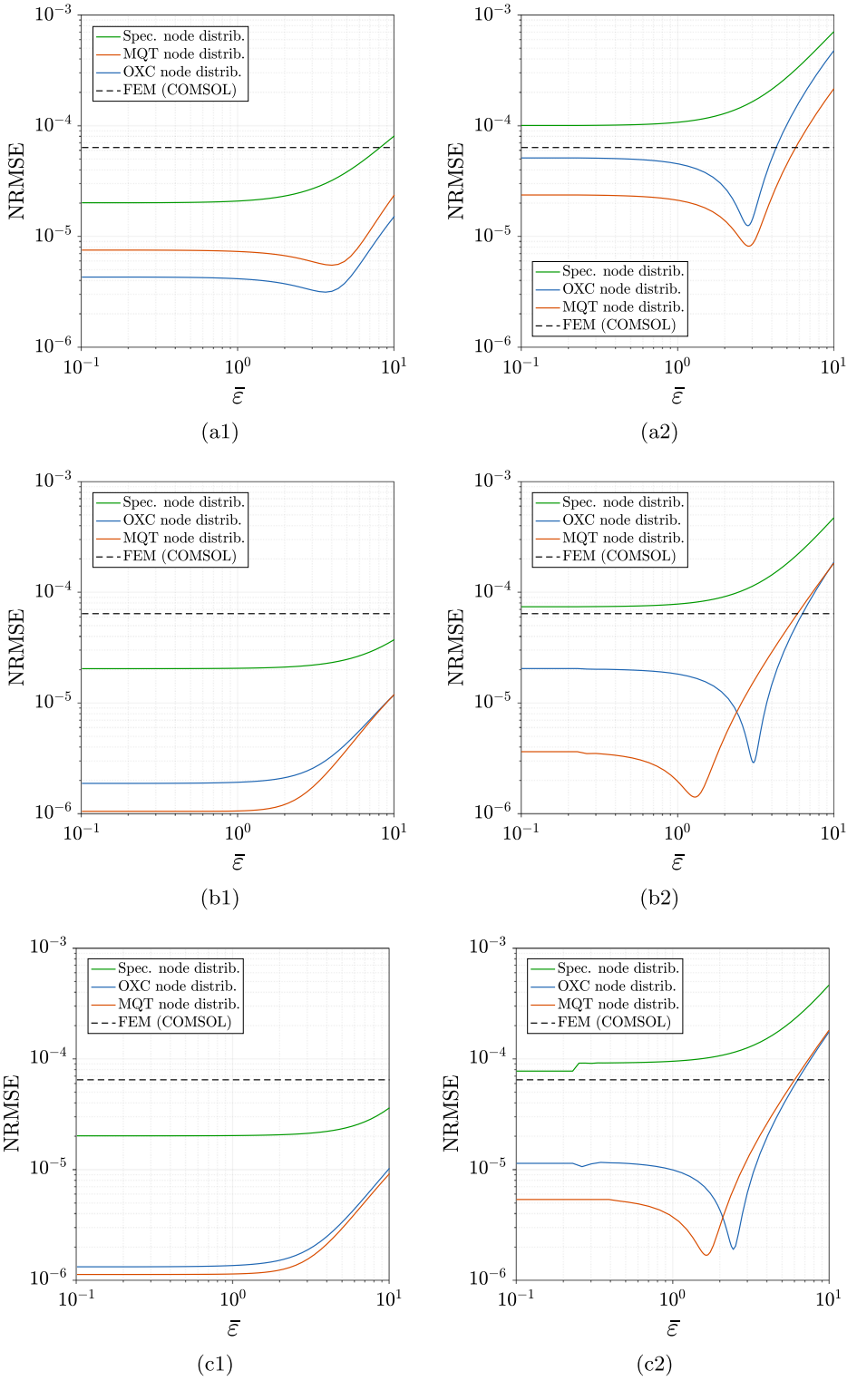

Fig. 5. Error vs rescaled shape factor $\bar{\varepsilon}$ for $n = 6$ (a), $n = 7$ (b) and $n = 8$ (c) neighboring nodes, test case A1 (1) and test case A2 (2), $N \approx 10\,000$ nodes.

which possesses unbounded radial derivatives in $r = 0$; the analytical solution (17) is also harmonic: $q = \nabla^2 T_{an} = 0$. The prescribed spacing function is chosen to be $s \propto r^{1/4}$ in order to adequately resolve the singularity in $r = 0$.

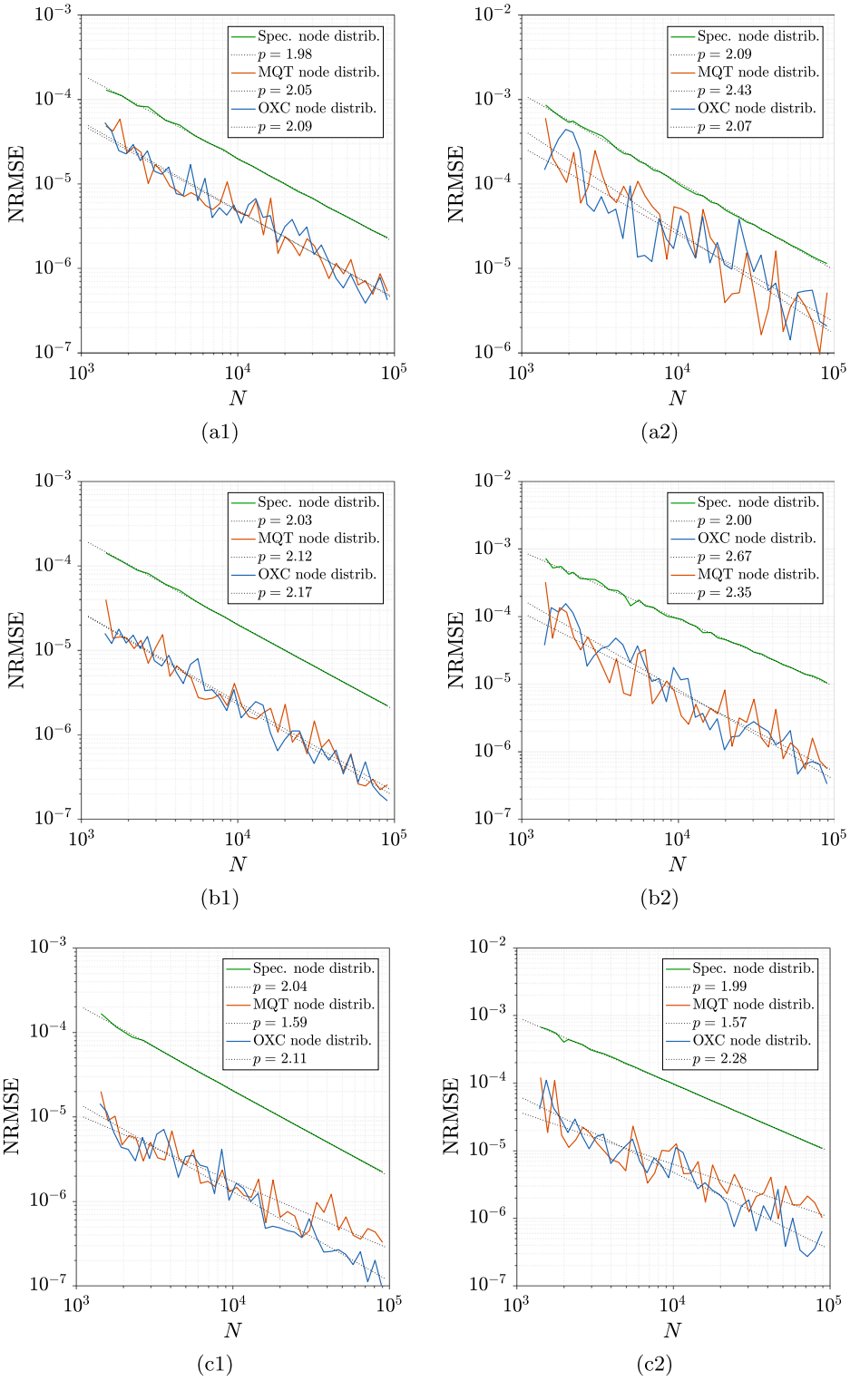

Fig. 6. Error vs number of nodes N for $n = 6$ (a), $n = 7$ (b) and $n = 8$ (c) neighboring nodes, test case A1 (1) and test case A2 (2), $\bar{\varepsilon} = 1$.

A specific and regular geometry dependent node distribution is reported in Fig. 7b, while Fig. 7c and 7d show the node distributions generated by the proposed approaches, OXC and MQT respectively; Dirichlet boundary conditions have been imposed along the whole boundary.

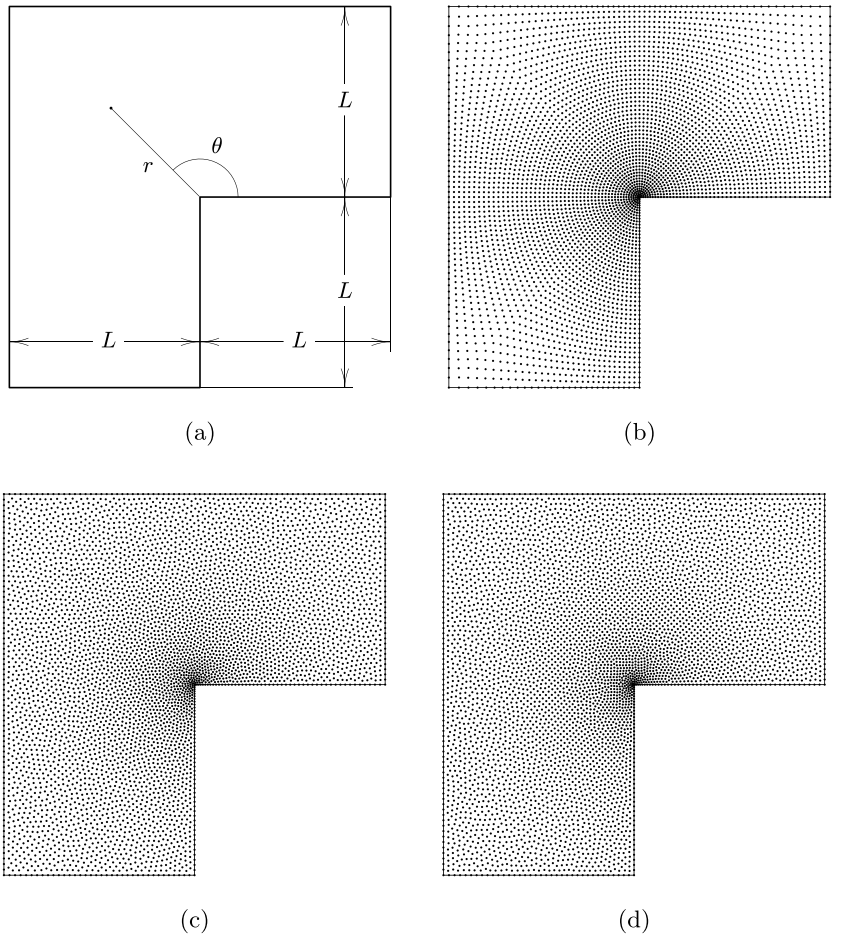

Fig. 7. Test case B, geometry (a) and node distributions: specific/regular (b), OXC (c) and MQT (d), $N \approx 5000$ nodes.

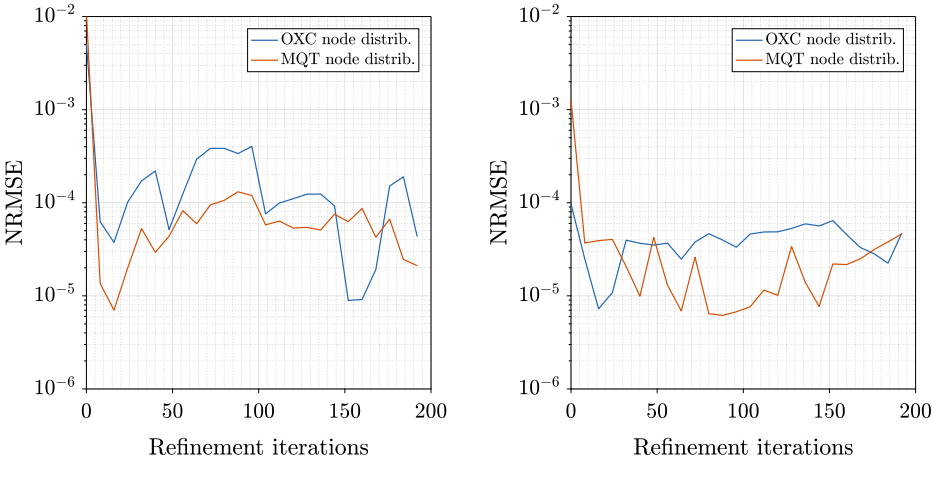

Fig. 8. Test case B: errors vs number of refinement iterations for $n = 6$ (left) and $n = 8$ (right) neighboring nodes, $N \approx 10\,000$ nodes and $\bar{\varepsilon} = 1$.

As in test case A, we first investigated the effect of the number of refinement iterations on the error for $n = 6$ and $n = 8$, Fig. 8, with $N \approx 10\,000$ nodes and $\bar{\varepsilon} = 1$. Except for the OXC node distribution for $n = 8$, it can be seen that, again, the use of the refinement algorithm is extremely effective in the error reduction within the first $15 \div 25$ iterations, while beyond

that point the error does not show monotonic behavior; for this reason we chose again a fixed number of 24 refinement iterations for every other result within this test case.

The results of sensitivity analysis with respect to the rescaled shape factor $\bar{\varepsilon}$ are reported in Fig. 9a for each number of local nodes $n = 6, 7, 8$ and $N \approx 10\,000$ nodes; FEM results are also displayed as reference.

From these figures it can be seen that within the considered range for $\bar{\varepsilon}$, it has negligible effect on the error for each type of node distribution, also because of the chosen type of analytical solution with a singularity.

However, as in test case A, it can be observed that errors obtained with the regular node distribution are again larger than the errors from the refined distributions (MQT and OXC), especially for $n = 7, 8$ neighboring nodes; in this case the regular node distribution here employed does not have optimal subdomain node arrangements with $n = 9$ nodes, but still, for $n = 7, 8$ the refined distributions are more effective and lead to smaller errors than FEM results for all cases. Since the rescaled shape factor does not affect the results, we chose $\bar{\varepsilon} = 1$ for the following results within this test case.

Fig. 9b shows the convergence curves for each number of local nodes $n = 6, 7, 8$ and $\bar{\varepsilon} = 1$. As observed from sensitivity analysis with respect to $\bar{\varepsilon}$, the regular node distribution produces greater errors than refined distributions and shows almost perfect 2nd order accuracy behavior for $n = 7, 8$, while for $n = 6$ all convergence curves show similar behavior; the refined distributions reveal also 2nd order accuracy for $n = 7, 8$, despite an irregular behavior as in the previous test case.

Again, from convergence curves for $n = 7, 8$ it can be desumed that the employment of refined distributions (MQT and OXC) can produce better results compared to regular distributions; finally, MQT and OXC node distributions show similar convergence curves in all cases.

The following points summarize the results for this test case:

- Again, a small number of node refinement iterations (≈ 24) is enough to reduce the final error by some orders of magnitude, for both OXC and MQT node distributions;
- Refined OXC and MQT node distributions always allow smaller errors than the regular node distributions, but the difference is less evident than test case A;
- An approximate $\approx$ 2nd order trend can be desumed for the accuracy;
- Shape factor influence is negligible in the range [0.1, 10] here considered;
- The error strongly depends upon the accurate solution near the singularity: the influence of the node distribution near the corner is stronger than in the remaining part of the domain;

4.3. Test case C: image defined spacing function

In order to show how the proposed node generation algorithms are also able to deal with different node density variations, we tested our generators with image defined spacing functions; given a square image defined on $\Omega = [0, 1]^2$ by RGB values of pixels ($R, G, B \in [0, 1]$) and their brightness ν defined as:

$$\nu = \frac{R + G + B}{3} \tag{18}$$

we define the spacing function s to be proportional to:

$$s \propto 1 + 2\nu^2 \tag{19}$$

while the reference solution T_{an} is here indirectly defined by the RHS of Eq. (4):

$$\nabla^2 T_{an} = q_{an} \tag{20}$$

defined over the square domain $\Omega = [0, 1]^2$ with zero valued Dirichlet boundary conditions:

$$T_{an}(\partial\Omega) = 0. \tag{21}$$

The RHS of Eq. (20) is then defined as:

$$q_{an} = (\eta - \min_{\Omega} \eta)^2 \tag{22}$$

where $\eta = 1 - \nu$ is the brightness complement; therefore the RHS term $q_{an} \geq 0$ increases where the reference image has low brightness while approaches 0 where the reference image reaches its maximum brightness: on the contrary the spacing function defined by Eq. (19) assumes small values (thick node distribution) in the former case and large values (coarse node distribution) in the latter case.

The reference solution T_{an} is then evaluated by solving Eq. (20) with a 2nd order FD discretization (5 points stencil) on a uniform 2048×2048 grid and an accurate solver based on matrix eigendecomposition.

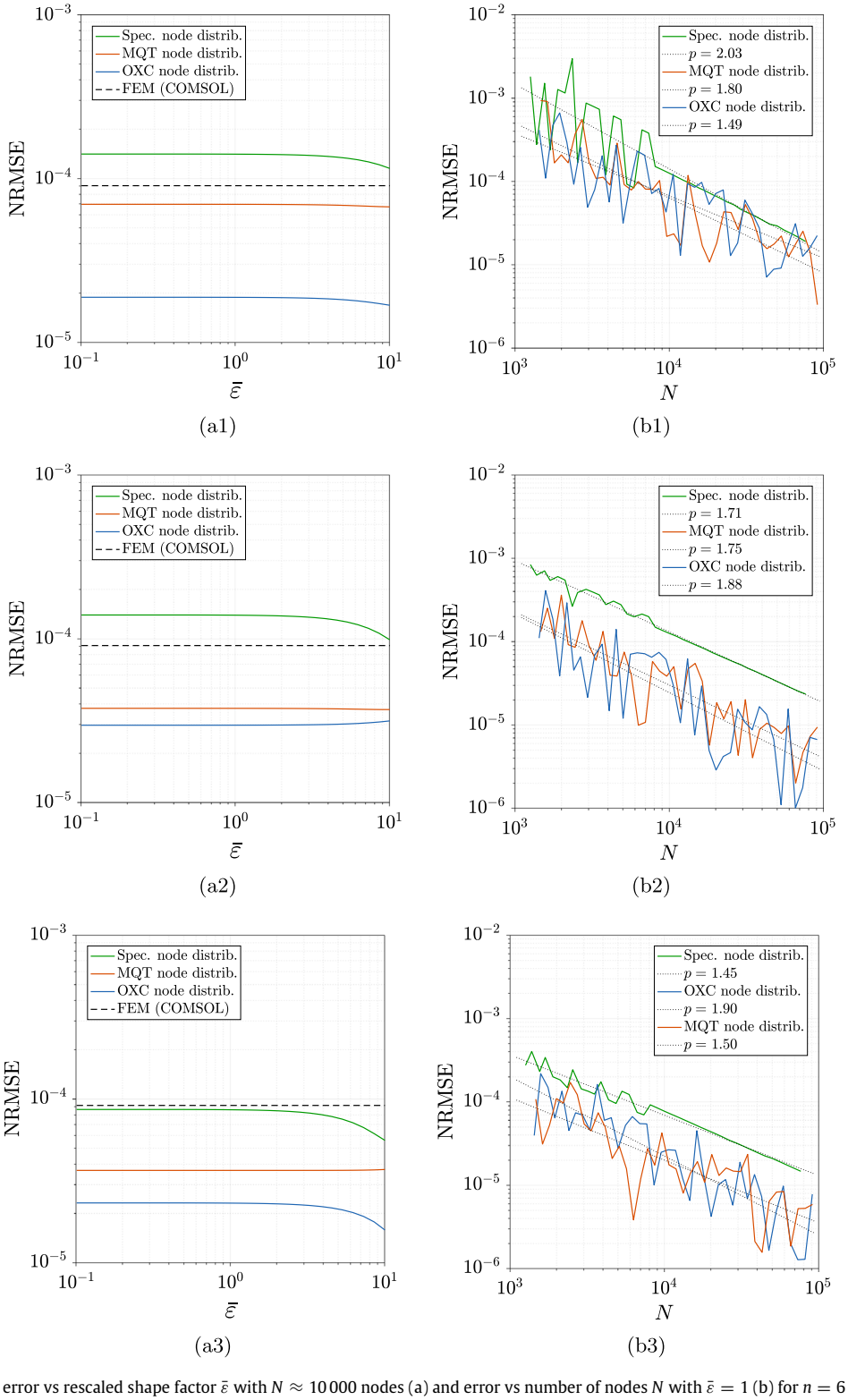

Fig. 9. Test case B: error vs rescaled shape factor $\bar{\varepsilon}$ with $N \approx 10\,000$ nodes (a) and error vs number of nodes N with $\bar{\varepsilon} = 1$ (b) for $n = 6$ (1), $n = 7$ (2) and $n = 8$ (3) neighboring nodes.

The classical Lena test image (512 × 512 pixels, grayscale) shown in Fig. 12a has been employed; Fig. 10 shows the corresponding node distributions generated by the proposed approaches, OXC and MQT respectively, with 24 node refinement iterations.

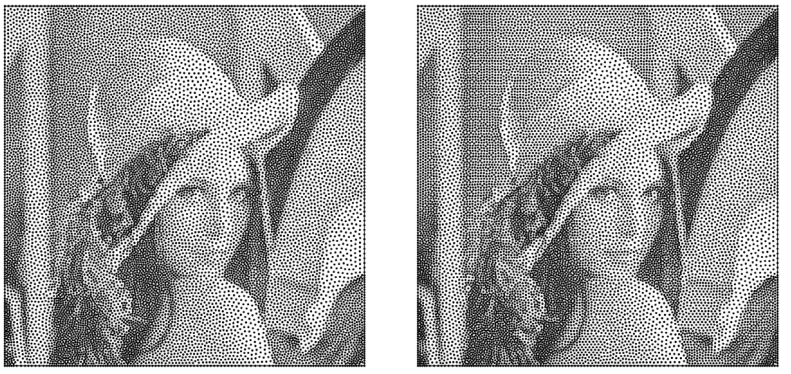

Fig. 10. Test case C, node distributions: OXC (left) and MQT (right) with $N \approx 15\,000$ nodes.

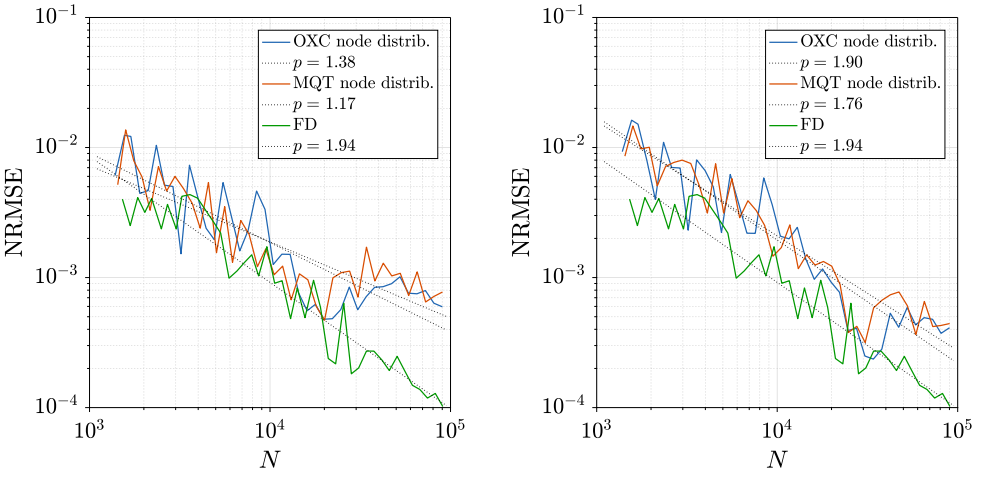

Fig. 11. Test case C: error vs number of nodes N for $n = 7$ (left) and $n = 8$ (right) neighboring nodes with $\bar{\varepsilon} = 1$.

Fig. 11 shows the convergence curves for $n = 7, 8$ neighboring nodes, a rescaled shape factor $\bar{\varepsilon} = 1$ and 24 node refinement iterations; convergence curves obtained with the above mentioned FD method are also reported as reference. All these convergence curves have irregular behavior, due to the irregular behavior of the image defined RHS term q_{an}; however, the curves from both meshless refined distributions (OXC and MQT) show similar behavior with FD curves with an approximatively overall 2nd order accuracy, while above $N \approx 30\,000$ nodes the meshless results exhibit some convergence issue, especially with $n = 7$. This could be due to the employment of a spacing function s with steep node density variations combined with an irregular behaved RHS term q_{an}; however, this is a limit case in the application of proposed node generation algorithms for meshless discretization of diffusion problems.

4.4. Test case D: stippling

This test case will not present any numerical result, but only some visual results of the application of the proposed algorithm for the halftone approximation of images through the stippling technique [19]. Stippling is employed to approximate the continue halftones of a grayscale image with an adequate distribution of equally sized black dots over a white background (or vice versa: white over black); it differs from the dithering techniques which are widely used in image processing [20] where the black dots can only have fixed positions and sizes (pixels).

From this point of view, the proposed node generation algorithms can also be used for stippling; the relation between the image brightness ν and the spacing function s is now needed. For simplicity we assume that the local brightness ν can be expressed as the complement of the ratio between the local area which is covered by the dots A_c and the local area A:

$$\nu = 1 - \frac{A_c}{A} = 1 - \frac{N_d A_d}{A} = 1 - \frac{A_d}{s^2} \tag{23}$$

Fig. 12. Test case D: halftone approximation of grayscale test image (a) with OXC (b) and MQT (c), $N \approx 50\,000$ nodes.

where N_d is the local number of dots, A_d is the dot area and $N_d/A = 1/s^2$ is a mean local approximation of spacing definition (1). Assuming circular dots with diameter d_d, from Eq. (23) we have:

$$s = d_d \sqrt{\frac{\pi}{4(1-\nu)}}. \tag{24}$$

A clear fact is that the smaller the dots, the smaller the spacing and therefore more and more dots will be needed for the approximation of a certain region; obviously, Eq. (24) does not account for overlapping dots.

Figs. 12a and 13a show two classical grayscale test images that have been considered for this problem, while Figs. 12b, 12c, 13b and 13c show their stippling approximation using both the proposed algorithms, namely OXC and MQT, with 24 node refinement iterations and $N \approx 50\,000$ nodes for each case.

Both the proposed algorithms produced images with remarkable quality and visually satisfactory node arrangements, without any artifact or noticeable patterns; from these visual tests it can also be observed how our procedures can effectively handle node densities with steep variations.

4.5. Implementation details and CPU times

The implementation of the presented procedures has been done using MATLAB® environment for the exchange and visualization of data and for the direct *LU* solution of sparse linear system (14), while the remaining part of the computational expensive tasks, such as the initial node placing phase, nearest neighbor search, iterative refinement phase and local matrix inversion (13) have been carried out through MATLAB linked MEX functions which are compiled from C source code. The initial node placing phase through OXC algorithm has a specific time consumption of 0.2 s/million nodes, 0.18 s/million nodes for the MQT algorithm, while the refinement phase takes approximately 0.6 s/(million nodes $\times$ iteration) on a modern laptop Intel® i7 2.6 GHz using only one single core. It is indeed evident that the refinement phase is the most expensive, since a

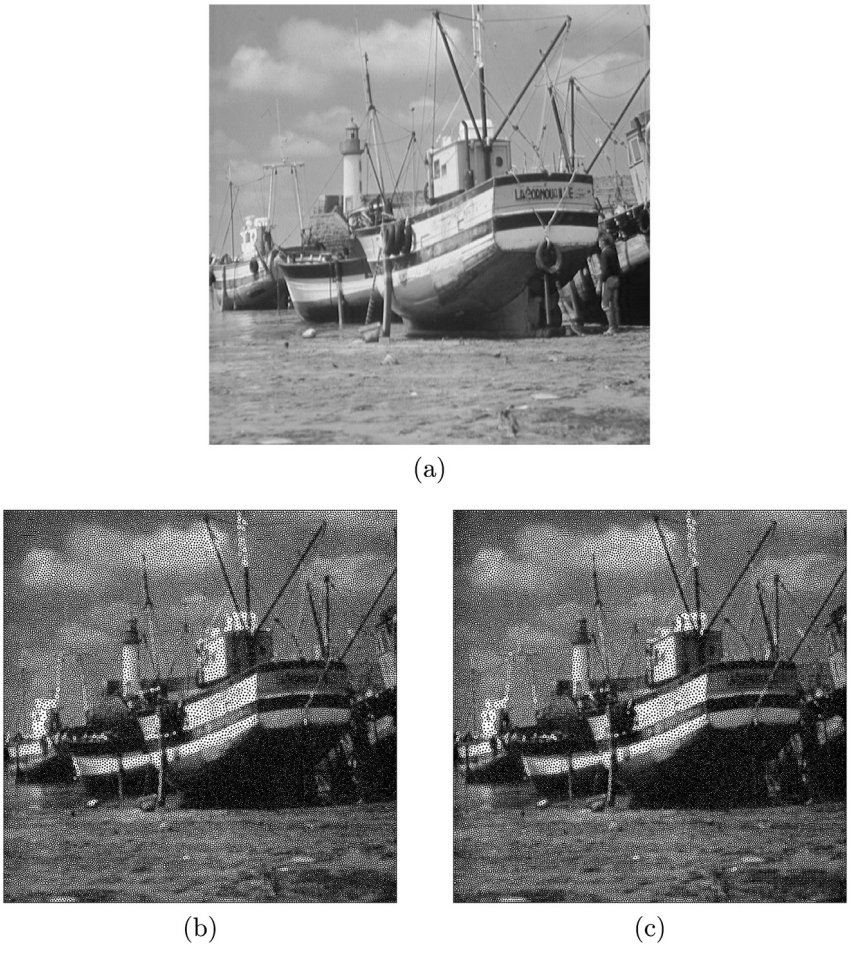

Fig. 13. Test case D: halftone approximation of grayscale test image (a) with OXC (b) and MQT (c), $N \approx 50\,000$ nodes.

minimum number of $10 \div 20$ iterations are always needed within the presented procedures in order to obtain suitable node arrangements. Nonetheless we point out the importance of the initial node placing phase through the proposed algorithms without which the refinement phase would need a much larger number of iterations.

4.6. Conclusions

Two algorithms for fast 2D node generation are proposed, each of which is composed of an initial phase, where nodes are placed according to a prescribed spacing function, and a refinement phase which improves the node distribution quality and local spacing isotropy.

Several test cases have been considered, where the proposed algorithms have been used to generate node distributions which are then employed for a local RBF discretization of diffusion problems and for image halftone approximation.

For the local RBF discretization test cases, the computed solutions have been compared to the corresponding analytical ones; the influence of the most important parameters such as number of refinement iterations, number of local neighbors n, shape factor ε, total number of nodes N and boundary conditions has been thoroughly investigated. For each test case an approximate 2nd order accuracy has been obtained, even when considering singular solutions or RHS with steep variations; a remarkable fact is that in many cases the employment of the node distributions generated by the proposed algorithms produced better results than regular/specific node distributions.

Visually remarkable results have also been obtained when both proposed algorithms are employed for the halftone approximation of grayscale images through the stippling technique.

Alongside the ability to deal with steep node density variations, these favorable properties confirm that the proposed algorithms can be promising candidates for 2D meshless node generation; the extension to 3D cases is straightforward for the MQT algorithm, while a 3D extension of the OXC algorithm would require some additional work (space partitioning), but is still possible.

References

[1] P.J. Frey, P.-L. George, A comprehensive survey of mesh generation methods, in: Mesh Generation, ISTE, London, UK, 2010, pp. 95–131. http://dx.doi.org/10.1002/9780470611166.ch3.

[2] S. Owen, A survey of unstructured mesh generation technology, in: 7th International Meshing Roundtable, 1998, 239–267. http://ima.udg.edu/~sellares/ComGeo/OwenSurv.pdf.

[3] H. Li, S. Mulay, Meshless Methods and Their Numerical Properties, CRC Press, 2013. http://dx.doi.org/10.1201/b14492. http://www.crcnetbase.com/doi/book/10.1201/b14492.

[4] B. Fornberg, N. Flyer, Solving PDEs with radial basis functions, Acta Numer. 24 (2015) 215–258. http://dx.doi.org/10.1017/S0962492914000130. http://www.journals.cambridge.org/abstract_S0962492914000130.

[5] S.A. Sarra, E.J. Kansa, Multiquadric radial basis function approximation methods for the numerical solution of partial differential equations, Adv. Comput. Mech. 2 (2009). http://www.scottsarra.org/math/papers/mqMonographSarraKansa.pdf.

[6] R. Loehner, E. Onate, An advancing front point generation technique, Commun. Numer. Methods. Eng. 14 (12) (1998) 1097–1108. http://dx.doi.org/10.1002/(SICI)1099-0887(199812)14:12<1097::AID-CNM183>3.0.CO;2-7. http://onlinelibrary.wiley.com/doi/10.1002/(SICI)1099-0887(199812)14:12%3C1097::AID-CNM183%3E3.0.CO;2-7/abstract.

[7] C.K. Lee, A new finite point generation scheme using metric specifications, Internat. J. Numer. Methods Engrg. 48 (10) (2000) 1423–1444. http://dx.doi.org/10.1002/1097-0207(20000810)48:10<1423::AID-ME948>3.0.CO;2-T. http://doi.wiley.com/10.1002/1097-0207%2820000810%2948%3A10%3C1423%3A%3AAID-NME948%3E3.0.CO%3B2-T.

[8] M.U. Varma, R.S. Rao, S. Deshpande, Point distribution generation using hierarchical data structures, Eccomas (2004) 1–6. http://www.mit.jyu.fi/eccomas2004/proceedings/pdf/276.pdf.

[9] B. Fornberg, N. Flyer, Fast generation of 2-D node distributions for mesh-free PDE discretizations, Comput. Math. Appl. 69 (7) (2015) 531–544. http://dx.doi.org/10.1016/j.camwa.2015.01.009. http://www.sciencedirect.com/science/article/pii/S0898122115000334.

[10] H. Samet, The quadtree and related hierarchical data structures, ACM Comput. Surv. 16 (2) (1984) 187–260. http://dx.doi.org/10.1145/356924.356930. https://dl.acm.org/citation.cfm?id=356930.

[11] P.J. Frey, P.-L. George, Quadtree-octree based methods, in: Mesh Generation, ISTE, London, UK, 2010, pp. 163–199. http://dx.doi.org/10.1002/9780470611166.ch5. http://doi.wiley.com/10.1002/9780470611166.ch5.

[12] R.W. Floyd, L. Steinberg, An Adaptive algorithm for spatial greyscale, Proc. Soc. Inform. Display 17 (2) (1976) 75–77. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0016965187{&}partnerID=40{&}md5=a83f53d39bde162bdc21e1c95d87deab.

[13] B. Šarler, R. Vertnik, Meshfree explicit local radial basis function collocation method for diffusion problems, Comput. Math. Appl. 51 (8 SPEC. ISS.) (2006) 1269–1282. http://dx.doi.org/10.1016/j.camwa.2006.04.013. http://www.sciencedirect.com/science/article/pii/S0898122106000836.

[14] B. Mavrič, B. Šarler, Local radial basis function collocation method for linear thermoelasticity in two dimensions, Internat. J. Numer. Methods Heat Fluid Flow 25 (6) (2015) 1488–1510. http://dx.doi.org/10.1108/HFF-11-2014-0359. http://www.emeraldinsight.com/doi/abs/10.1108/HFF-11-2014-0359.

[15] R.L. Hardy, Multiquadric equations of topography and other irregular surfaces, J. Geophys. Res. 76 (8) (1971) 1905–1915. http://dx.doi.org/10.1029/JB076i008p01905. http://doi.wiley.com/10.1029/JB076i008p01905.

[16] N. Flyer, B. Fornberg, V. Bayona, G.A. Barnett, On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy, J. Comput. Phys. 321 (2016) 21–38. http://dx.doi.org/10.1016/j.jcp.2016.05.026. http://www.sciencedirect.com/science/article/pii/S0021999116301632.

[17] V. Bayona, N. Flyer, B. Fornberg, G.A. Barnett, On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs, J. Comput. Phys. 332 (2017) 257–273. http://dx.doi.org/10.1016/j.jcp.2016.12.008. http://www.sciencedirect.com/science/article/pii/S0021999116306490.

[18] A. Plaza, M.C. Rivara, On the adjacencies of triangular meshes based on skeleton-regular partitions, J. Comput. Appl. Math. 140 (1–2) (2002) 673–693. http://dx.doi.org/10.1016/S0377-0427(01)00484-8. http://www.sciencedirect.com/science/article/pii/S0377042701004848.

[19] D. Martín, G. Arroyo, A. Rodríguez, T. Isenberg, A survey of digital stippling, Comput. Graph. (Pergamon) 67 (October) (2017) 24–44. http://dx.doi.org/10.1016/j.cag.2017.05.001. http://www.sciencedirect.com/science/article/pii/S0097849317300432.

[20] R. Ulichney, A review of halftoning techniques, Proc. SPIE - Int. Soc. Opt. Eng. 3963 (2000) 378–391. http://dx.doi.org/10.1117/12.373419. http://adsabs.harvard.edu/abs/1999SPIE.3963..378U.