Computer Physics Communications 180 (2009) 1125–1133

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

A generalized finite difference method using Coatmèlec lattices

Miguel A. García-March [a,*], Miguel Arevalillo-Herráez [b], Francisco R. Villatoro [c], Fernando Giménez [a], Pedro Fernández de Córdoba [a]

[a] Interdisciplinary Modeling Group, InterTech., Institut Universitari de Matemàtica Pura i Aplicada – IUMPA, Universitat Politècnica de València, 46022 València, Spain
[b] Departament d'Informàtica, Universitat de València, Avda. Vicente Andrés Estellés, 1, 46100 Burjasot, Spain
[c] Departamento de Lenguajes y Ciencias de la Computación, E.T.S.I. Industriales, Universidad de Málaga, Campus El Ejido, s/n, 29013 Málaga, Spain

ARTICLE INFO

Article history:
Received 17 September 2007
Received in revised form 24 September 2008
Accepted 19 January 2009
Available online 22 January 2009

Keywords:
Generalized finite difference method
Irregular meshing
Mesh generation
Coatmèlec lattices

ABSTRACT

Generalized finite difference methods require that a properly posed set of nodes exists around each node in the mesh, so that the solution for the corresponding multivariate interpolation problem be unique. In this paper we first show that the construction of these meshes can be computerized using a relatively simple algorithm based on the concept of a Coatmèlec lattice. Then, we present a generalized finite difference method which provides a numerical solution of a partial differential equation over an arbitrary domain, using the generated meshes. The accuracy and mesh adaptivity of the method is evaluated using elliptical equations in several domains.

1. Introduction

The numerical methods for the solution of partial differential equations over irregular domains, such as finite differences or finite elements, are characterized by the definition of the shape functions used for the calculation of the derivatives of the unknown function by interpolation. In meshless methods, the shape functions depend only on the nodes positions and the nodal connectivity, instead of being fixed for all the nodes of the mesh as in standard techniques [1]. The list of methods referred to as meshless is very large and it is continuously growing. For example, smooth particle hydrodynamics [2], generalized finite differences [3,4], moving least squares techniques [5], diffuse elements [6], element-free Galerkin [7], to name only a few (a more comprehensive list can be found in [1]). Meshless methods are useless without an evaluation of the nodal connectivity bounded in time and a computational cost which grows linearly with the total number of nodes in the domain [1]. Here, a new generalized finite difference method is introduced having such a property.

Generalized finite difference methods (GFDMs) can be applied when either the domain, the distribution of nodes, or both, are non-rectangular or irregular [8–10]. A major difficulty in their development is that, to find the coefficients of the finite difference formulae at some nodes of the mesh, a linear system of equations whose coefficient matrix may be singular needs to be solved [11–14]. To avoid this problem two main alternatives have been proposed: (i) the use of polynomial fitting instead of polynomial interpolation, as in moving least squares techniques, resulting in over-determined systems of equations [3,12,15–17]; and (ii) the generation of a mesh with a structure such that the finite difference stencils around each node always yield a square system of linear equations which has a unique solution. The latter approach has received little attention in the literature.

In the context of multivariate Lagrange interpolation, there is an extensive research on the construction of distributions of nodes which assure the existence and uniqueness of the interpolant. These are called properly posed set of nodes (PPSNs) [18–20]. Among the simplest PPSNs in the plane are Coatmèlec lattices [19,21,22], which also can be extended easily to arbitrary number of dimensions [21].

In this paper, a new mesh generation algorithm based on a Coatmèlec distribution of nodes is introduced, and applied in the context of a GFDM for linear partial differential equations. The resulting numerical method allows the control of the order of accuracy and can be applied to problems whose domain is irregularly meshed.

The remaining of this paper is organized as follows. The problem of numerical differentiation on irregular meshes is introduced in Section 2 for further reference in the rest of the paper. Section 3 presents a method to generate meshes which are suitable for GFDMs. The stars, which are Coatmèlec lattices in such meshes, are obtained using the algorithms presented in Section 4, whose computational cost is also briefly analyzed. Section 5 discusses the properties of the shape functions corresponding to the stars. Rep-

* Corresponding author. Tel.: +34 963 877 662, ext. 88395.
E-mail address: migarma1@doctor.upv.es (M.A. García-March).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2009.01.015

resentative results of the accuracy of the GFDMs in these meshes are provided in Section 6, where the adaptivity of the new algorithm is also explored. Finally, the main conclusions and several future lines of research work are discussed.

2. Numerical differentiation on irregular meshes

GFDMs for partial differential equations replace the continuous partial derivatives in the equation by numerical differentiation formulae based on a polynomial interpolation on a set of nodes. These formulae are obtained as the solution of a linear system of equations. The problem is that, depending on the nodes chosen, this system may be singular. In this paper we show how the set of nodes can be chosen so that a non-singular system results.

Let us recall the basics of two-dimensional numerical differentiation. Let $f(x,y) \in C^{m+1}(\Omega)$ be a function with $(m+1)$ continuous derivatives in a domain $\Omega \subset \mathbb{R}^2$ and take a lattice (set of non-repeated nodes) $X_m = \{w_i\}$, $i = 1, 2, \ldots, N$, where $w_i \equiv (x_i, y_i) \in \Omega$ and N is arbitrary. A polynomial $p_m(x,y)$ of degree at least m, written in the canonical basis as

$$p_m(x,y) = \sum_{j=0}^{m} \sum_{l=0}^{j} x^l y^{j-l} \alpha_{jl}, \quad \alpha_{jl} \in \mathbb{R},$$

interpolates the function $f(x,y)$ on X_m if $p_m(w_i) = f(w_i)$, $i = 1, 2, \ldots, N$. So the coefficients $\{\alpha_{jl}\}$ are the solution of the linear system of equations given by

$$\sum_{j=0}^{m} \sum_{l=0}^{j} x_i^l y_i^{j-l} \alpha_{jl} = f(w_i), \tag{1}$$

whose coefficient matrix is square only if $N = (m+1)(m+2)/2$, being in such a case a bivariate Vandermonde matrix. An interpolation problem is *posed* with respect to a lattice X_m if the corresponding bivariate Vandermonde matrix is non-singular (the linear system has a unique solution) for every function $f(x,y) \in C^{m+1}(\mathbb{R}^2)$.

The (r,s)th partial derivative of the function $f(x,y)$ may be approximated at a point, say w_q, inside the convex hull of X_m, by differentiation of the corresponding interpolating polynomial. This approximation results in a linear combination of the values of the function at the nodes of X_m, the so-called *star*, *stencil*, or *cloud* of the node, i.e.,

$$\begin{aligned} \frac{\partial^{r+s} f(w_q)}{\partial x^r \partial y^s} &= \frac{\partial^{r+s} p_m(w_q)}{\partial x^r \partial y^s} + O\left(h_q^k\right) \\ &= \sum_{i=1}^{N} \beta_i^q f(w_i) + O\left(h_q^k\right), \end{aligned} \tag{2}$$

where $k = m + 1 - r + s$ is the order of accuracy and $h_q = \max_{i=2}^{N} \|w_i - w_q\|_\infty$ is the star diameter, $\|w_i - w_q\|_\infty = \max\{|x_i - x_q|, |y_i - y_q|\}$. The coefficients β_i^q are the solution of a linear system of equations whose coefficient matrix is the so-called star matrix.

The star matrix may also be determined by the method of undetermined coefficients. Without loss of generality the node w_q is translated to the origin $(0,0)$, hence $w_i = h_q u_i$, $i = 2, \ldots, N$, with $\|u_i\| \leqslant 1$, so that the Taylor series expansion of $f(w_i)$ around the origin, up to the order $m = r + s + k$, is given by

$$f(w_i) = f(0,0) + \sum_{t=1}^{m} h_q^t \sum_{j=0}^{t} \frac{\hat{x}_i^j \hat{y}_i^{t-j}}{j!(t-j)!} \frac{\partial^t f(0,0)}{\partial x^j \partial y^{t-j}} + O\left(h_q^{m+1}\right), \tag{3}$$

where $u_i = (\hat{x}_i, \hat{y}_i)$. The N coefficients β_i in Eq. (3) which approximate the (r,s)th partial derivative of the function are the solution of the system of N linear equations obtained by substitution of Eq. (3) into Eq. (2) and equating the result to that partial derivative, resulting in

$$\sum_{i=1}^{N} \hat{x}_i^j \hat{y}_i^{t-j} \beta_i = \frac{j!(t-j)!}{h_q^t} \delta_{s,j} \delta_{r,t-j}, \tag{4}$$

for all $0 \leqslant j \leqslant t \leqslant m$, where $\delta_{s,j}$ is the Kronecker delta, i.e., $\delta_{s,j} = 1$, if $s = j$, and $\delta_{s,j} = 0$, otherwise. Note that the star matrix is a bivariate Vandermonde independent of the degree (r,s) of the derivative, so the LU factorization is recommended for the numerical solution of Eq. (4) with different nonhomogeneous terms when several derivatives must be calculated.

3. Mesh generation

A mesh such that the stars around each node always yield a unique solution for either Eq. (1) or Eq. (4) requires that a PPSN exists around every node in the mesh. Such meshes cannot be produced using standard mesh generation techniques.

It has been proven that Coatmèlec lattices are PPSNs [23]. In this paper we make use of this fact to build an algorithm to generate meshes that are suitable for GFDMs. Below, we recall the definition of a Coatmèlec lattice.

Definition. Let $m \in \mathbb{N}$ and $N = (m+1)(m+2)/2$, then $X_m = \{w_i\}_{i=1}^{N}$ is a Coatmèlec lattice if there are $m+1$ disjoint sets of aligned nodes $\gamma_1, \gamma_2, \ldots, \gamma_{m+1}$ such that γ_m contains m nodes.

From the definition of a Coatmèlec lattice and the proof provided in [23], it is clear that a mesh suitable for GFDMs can be easily generated placing the nodes over strategically positioned lines. This method allows the generation of meshes for practically any domain, with the property that a Coatmèlec lattice around each node can be found. Fig. 1 shows some examples of meshes generated using this procedure (only a square zoom is shown in order to highlight the lines used in their construction). In Fig. 1(a), two sets of parallel and equally spaced lines are used, placing the nodes at their intersections. In Fig. 1(b), a less regular distribution is obtained perturbing the slopes of the lines slightly. In Figs. 1(c) and 1(d), irregular meshes are produced using random lines and locating the nodes at their intersections (c) or along the lines (d).

4. Algorithm for finding Coatmèlec lattices

To use a GFDM on an irregular mesh, it is necessary to find a star around each node and approximate every derivative in the partial differential equation to a given order of accuracy. In our method, this star is a Coatmèlec lattice composed of N nodes, with $N = (m+1)(m+2)/2$ and $m = k + s + r$. This section describes the method we have used to find the Coatmèlec star around each node w_q in a mesh generated as explained in the previous section.

Finding a star around a node w_q which satisfies the conditions to be a Coatmèlec lattice requires searching for $m+1$ disjoint sets of nodes $\{\gamma_1, \ldots, \gamma_{m+1}\}$, each set γ_i containing i aligned nodes from its neighborhood. Figs. 2 and 3 show examples of such stars for the second order derivative to $O(h_q)$ and $O(h_q^3)$ approximations, respectively, in each of the meshes presented in Fig. 1. These stars have been obtained using Algorithm 1, which presents the pseudo-code for the function `FindCoatmelecStar`.

In order to find the Coatmèlec lattice with `n` lines around a given node w_q, a set `Wq` containing the nodes in its local neighborhood is built, limiting the search to a circular region of a radius chosen according to the mesh density. Then, the function `FindCoatmelecStar` is invoked by the following initial call `Result ← FindCoatmelecStar(Wq,n)`. The output of the algorithm is a list of n sets of aligned nodes, i.e.,

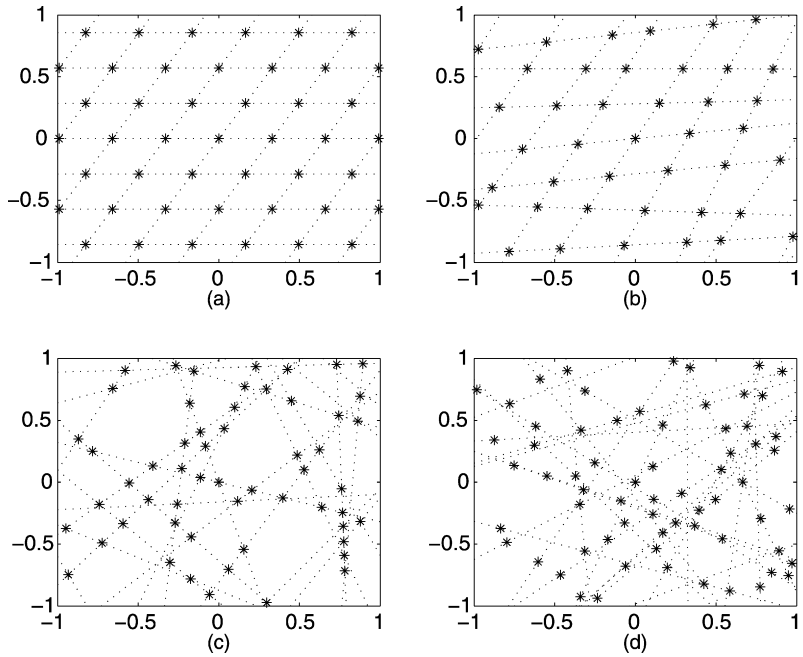

Fig. 1. Square zoom of meshes generated by lines obtained: (a) using two sets of parallel and equally spaced lines, (b) randomly perturbing the slopes of the lines in (a), and (c) and (d) using random lines. The nodes are located at the intersections between the lines in plots (a), (b), and (c); and randomly along the lines in (d).

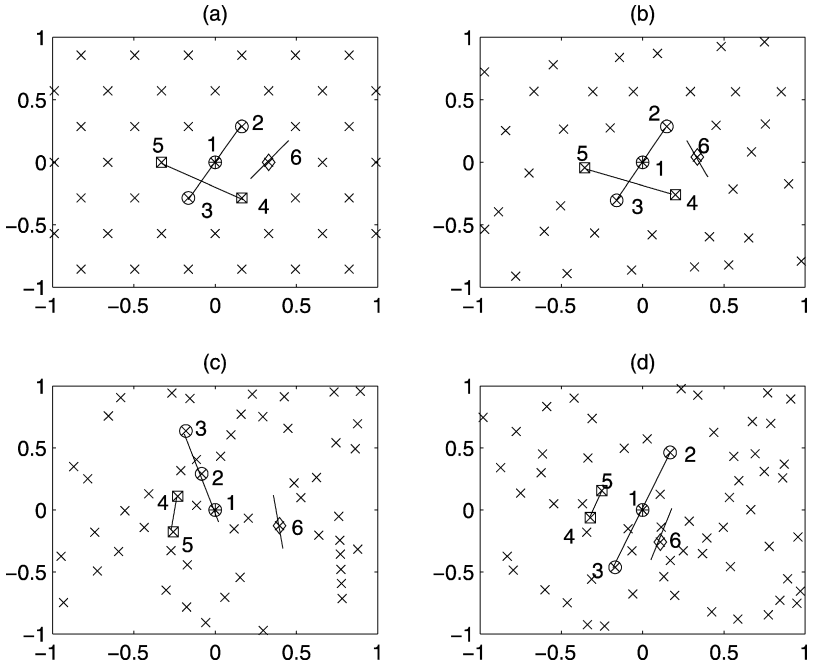

Fig. 2. Coatmèlec lattice in the meshes of Fig. 1 for a second order derivative and a first-order approximation. Nodes of the mesh are pictured by crosses. The 3 nodes in γ_2 are represented with circles, the 2 nodes in γ_1 with squares and the one in γ_0 with a diamond.

$\langle\{w_{n,1}, w_{n,2}, \ldots, w_{n,n}\}, \{w_{n-1,1}, w_{n-1,2}, \ldots, w_{n-1,n-1}\}, \ldots, \{w_{1,1}\}\rangle$, where $w_{i,j}$ is the jth node in the ith line, with $1 \leqslant j \leqslant i$, defining the Coatmèlec lattice.

Algorithm 1 uses a backtracking or depth-first procedure [24] to go through all the possible candidates, growing the solution one set at a time in a recursive fashion, using the `findAlignedNodes` function. At each step in the backtracking algorithm, we start from a given partial solution, say, $\langle\{w_{n,1}, w_{n,2}, \ldots, w_{n,n}\}, \{w_{n-1,1}, w_{n-1,2}, \ldots, w_{n-1,n-1}\}, \ldots, \{w_{j,1}, w_{j,2}, \ldots, w_{j,j}\}\rangle$. Then, we try to extend this partial solution by adding another set with $j-1$ nodes to this list. If this is possible, the algorithm recurs and continues. Otherwise, the last set from the partial solution, i.e., $\{w_{j,1}, w_{j,2}, \ldots, w_{j,j}\}$, is removed, and another possibility for it is attempted. The algorithm stops when a partial solution is also a complete solution or when all possibilities have been tried and a Coatmèlec lattice in the set `Wq` could not be found. Note that in Algorithm 1, the operator $a \ominus b$ where a and b are arrays of nodes, refers to the nodes which are in a but not in b.

Algorithm 1 uses the function `findAlignedNodes` defined in Algorithm 2. This function is invoked by `Result ← findAlignedNodes(pivots,Vq)`, where `pivots` is a pair of nodes in the neighborhood of w_q and `Vq` is the set of nodes to be processed (those which are not part of a line which is already an element of the solution). Its output are all nodes in `Vq` which be-

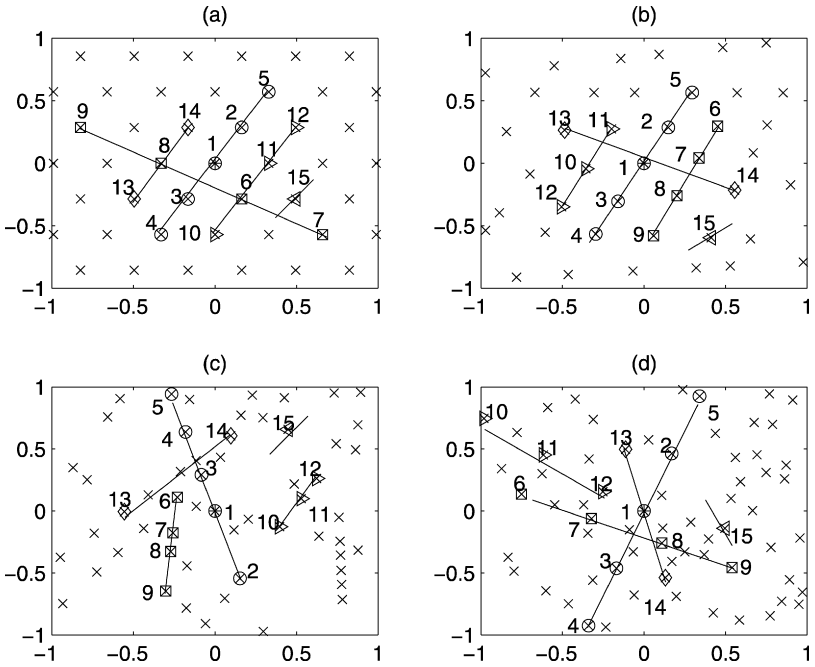

Fig. 3. Coatmèlec lattice in the meshes of Fig. 1 for a second-order derivative and a third-order approximation. Mesh nodes are represented with crosses. The 5 nodes in γ_4 with circles, the 4 nodes in to γ_3 with squares, the 3 nodes in γ_2 with the less than symbol, the 2 nodes in γ_1 with diamonds and the node in γ_0 with the greater than symbol.

Table 1
Coefficients β_i^q for the stars in Fig. 2 (a)–(d) for the first-order approximation to the second-order derivative.

i	(a)	(b)	(c)	(d)
1	−1.00	−1.06	−7.25	−3.44
2	0.00	0.01	4.05	0.80
3	0.00	−0.15	−1.07	−1.45
4	0.00	0.17	0.59	3.13
5	0.50	0.59	1.81	−1.44
6	0.50	0.45	1.87	2.40

Table 2
Coefficients β_i^q for the stars in Fig. 3 (a)–(d) for the third-order approximation to the second-order derivative.

i	(a)	(b)	(c)	(d)
1	−10.50	−6.46	−10.37	54.13
2	0.39	−0.25	22.73	−2.54
3	0.00	−0.20	2.15	7.24
4	0.00	−0.01	18.05	0.54
5	0.39	0.59	−3.31	−0.88
6	1.75	1.17	−72.06	6.26
7	5.25	3.16	128.06	−38.11
8	0.10	−0.16	−83.69	2.97
9	0.19	−0.68	7.69	−0.52
10	5.25	0.72	−11.41	−2.19
11	−0.29	3.30	21.11	14.02
12	−0.58	−0.61	−9.99	−43.44
13	0.00	−0.31	1.59	−1.58
14	−0.58	−0.45	−15.38	3.52
15	−1.17	0.22	4.82	0.60

long to the line defined by the pivots. The implementation of this function is straightforward using a function `isAligned` to check if a node belongs to the line defined by the two pivot nodes.

Once a star in `Wq` is found, the coefficients β_i^q are obtained solving the linear system of Eqs. (4). For illustration purposes, the finite difference coefficients for the stars shown in Figs. 2 and 3 are presented in Tables 1 and 2, respectively.

A desirable feature of this technique is that it is structurally consistent, in the sense that a zoomed view of a small area in a high density mesh presents exactly the same structure as a larger area from another lower density mesh. The effect of increasing the number of nodes in the accuracy of the result is determined by the order of accuracy imposed. Doubling the number of nodes in the mesh reduces the error by a factor of 2^ρ, with ρ the order of accuracy chosen. This effect will be shown experimentally in Section 6.

Let us present a brief analysis of the computational cost associated with Algorithm 1. In our current implementation, the time required to calculate a star around a node depends on the order of accuracy desired, the size of the local neighborhood considered (the number of nodes in the set W_q) and the disposition of the nodes. For a fixed order of accuracy, it is easy to establish a constant upper bound for the time required to calculate a star around a node (that required in the worst case). Therefore, the total computational cost for the assembly of the global stiffness matrix is $O(T)$, with T the total number of nodes in the mesh.

A comparison between our algorithm and classical Delaunay triangulation algorithms, e.g. [25,26], which have a computational cost $O(T \cdot \log T)$ [27], shows that, independently of the order of accuracy and the size of the neighborhood considered, it is always possible to find a threshold size such that our algorithm perform better than such techniques. In practice, using the current implementation of our method, it would only outperform Delaunay triangulation based algorithms for a low order of accuracy and a large number of nodes.

On this aspect, it is worth mentioning that from Lawson's incremental insertion algorithm for Delaunay triangulation [28], related research has been very intense and many efforts have focused on the optimization and refinement of the original method. In this respect, our algorithm has been implemented in Matlab and some further improvements need to be studied. In fact, a clear advantage of the technique proposed here is that it can be easily parallelized, since the algorithm for finding the stars around each node can be

```
Input:
Wq: array containing the nodes in the neighborhood of w_q
n: number of sets of aligned nodes to be found
Output:
Result: list of arrays of decreasing length which represent the solution, or an
empty list if one cannot be found
begin
  if (n = 1)
  then
    if Length (Wq)>0
    then
      Solution ← InsertInList (Solution, {Wq(1)} )
      return Solution
    end
  else
    indexFirstPivot ← 1
    indexSecondPivot ← 2
    while (indexFirstPivot<Length (Wq) )
    do
      pivots ← {Wq(indexFirstPivot),Wq(indexSecondPivot)}
      alignedNodes ←FindAlignedNodes (pivots,Wq)
      foreach possible subset Wn of n nodes in alignedNodes
      do
        Result ← FindCoatmelecStar (Wq○Wn, n-1)
        if (Result ≠ {})
        then return InsertInList (Result,Wn)
      end
      indexSecondPivot ← indexSecondPivot+1
      if (indexSecondPivot>Length(Wq))
      then
        indexFirstPivot ← indexFirstPivot+1
        indexSecondPivot ← indexFirstPivot+1
      end
    end
  end
  return {}
end
```

Algorithm 1. Result ← FindCoatmelecStar(Wq, n).

```
Input:
pivots: Array of two nodes
Wq: Array containing the nodes to be processed
Output:
Result: Set of aligned nodes, or an empty set if a solution is not found
begin
  Result ← {}
  foreach i ∈ [1..Length (Wq)]
  do
    if isAligned (Wq(i), pivots)
    then Result ← InsertInSet ( Result, Wq(i) )
  end
  return Result
end
```

Algorithm 2. Result ← FindAlignedNodes(pivots, Wq).

performed locally, thus being a good candidate for data partitioning.

Finally, let us note that the sparsity of the resulting stiffness matrix is similar to that of the matrix for classical Delaunay triangulation algorithms, hence the computational cost required for the solution of the resulting linear system of equations are equivalent between both techniques.

5. Shape functions

In generalized finite difference methods, the shape functions change from node to node, corresponding to the two-dimensional Lagrange polynomials defined by the nodes used in the stencil, i.e., two-dimensional polynomials with a value of unity at the expanding node and zero in the other ones. For the Coatmèlec lattices,

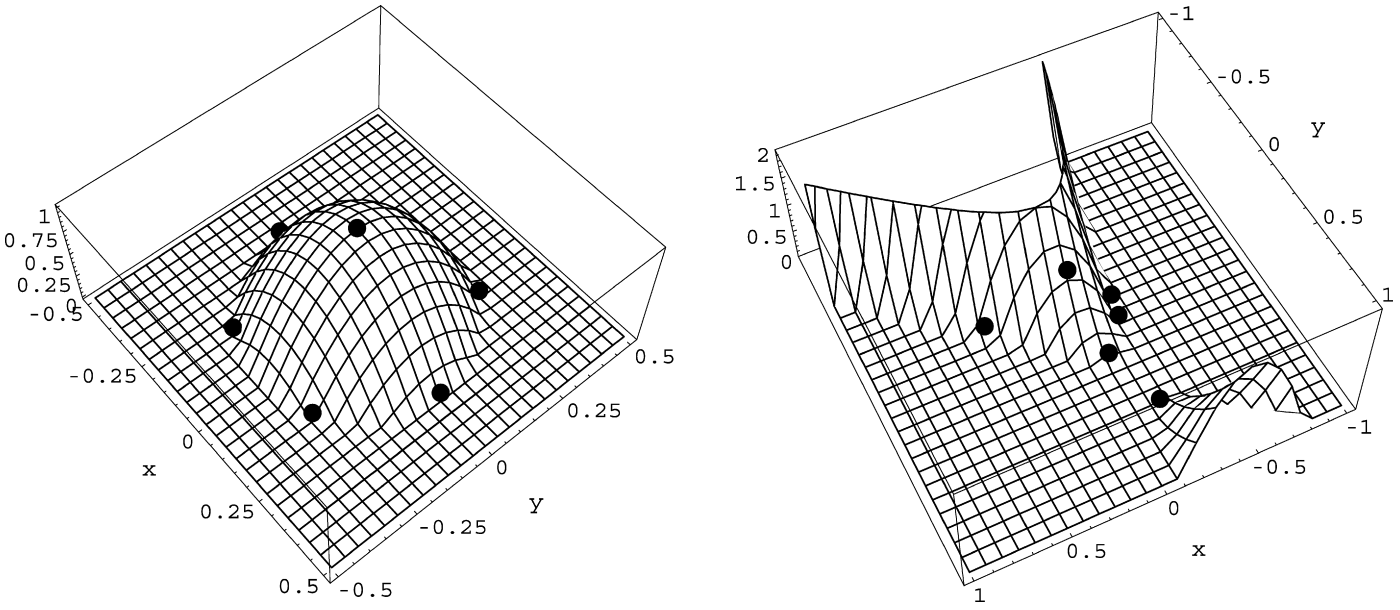

Fig. 4. Shape functions for the stencils in Figs. 2(a) (left plot) and 2(c) (right one). The black dots indicate the position of the nodes.

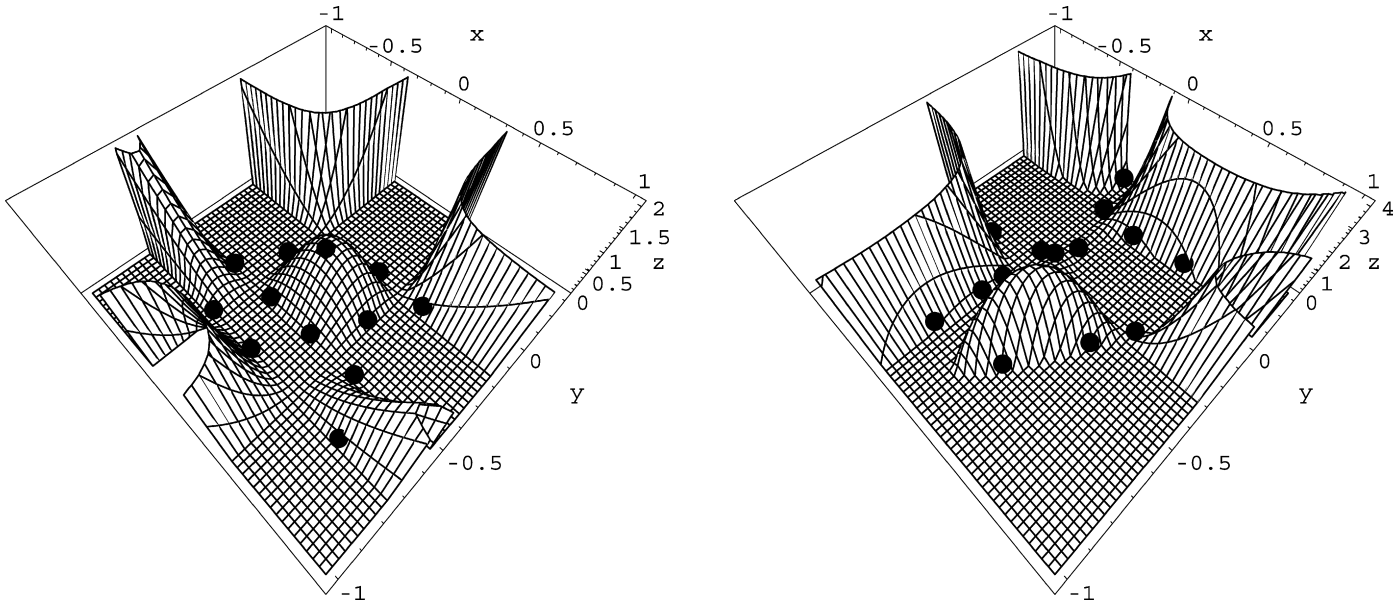

Fig. 5. Shape functions for the stencils in Figs. 3(a) (left plot) and 3(c) (right one). The black dots indicate the position of the nodes.

Gasca and Maeztu [29] have obtained a closed form expression for these polynomials using the Newton form by means of a recursive evaluation, cf. Eqs. (21)–(29) in Ref. [29]. For brevity, this expression is omitted here. In fact, up to the authors' knowledge, there is no easier way to write such Lagrange polynomials.

The "outlook" of the shape functions at each node depends strongly on the stencil, hence they are usually not explicitly presented in the papers dealing with generalized finite difference methods. Figs. 4 and 5 show the shape functions of the stencils in Figs. 2(a) and 2(c), and Figs. 3(a) and 2(c), respectively. Note that only positive values of the corresponding polynomials have been plotted in the figures. The shape function has value one at each node. This may be either a local maximum, as in the left plots in Figs. 4 and 5, or not, as in the right ones in Figs. 4 and 5.

The shape functions for the stencils in Fig. 2 (shown in Fig. 4) are quadratic, two-dimensional polynomials whose first derivatives are piecewise linear functions and whose second derivatives are piecewise constant. Similarly, the shape functions for the stencils in Fig. 3 are cubic, two-dimensional polynomials. Fig. 5 shows that such polynomials grow very fast even near the nodes of the Coatmèlec lattice, resulting in large values for their derivatives. Plots showing the first and second derivatives of the shape functions have been omitted here for the sake of brevity.

6. Presentation of results

Let us apply a generalized finite difference method based on the algorithm developed in this paper to the solution of the elliptic differential equation

$$\begin{aligned} &\Delta F(x,y) = 0, && (x,y) \in \Omega, \\ &F(x,y) = g(x,y), && (x,y) \in \partial\Omega, \end{aligned} \tag{5}$$

where Δ is the Laplacian and F a scalar function.

Let us take the two domains shown in Fig. 6:

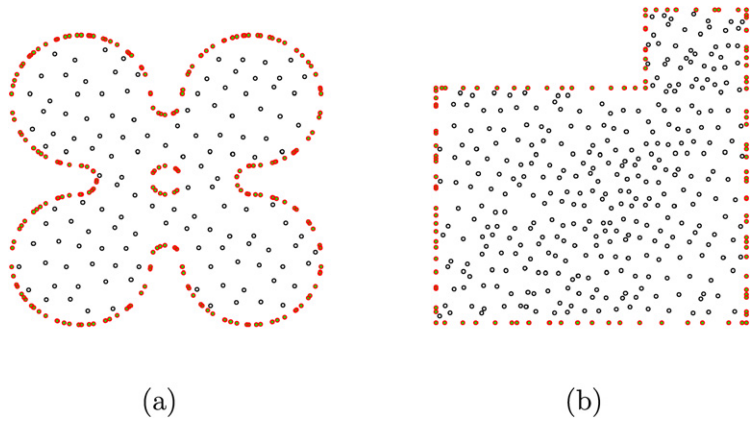

Fig. 6. Examples of irregular meshes defined on the two domains described in the text: (a) Four-leaf clover, and (b) domain with corners.

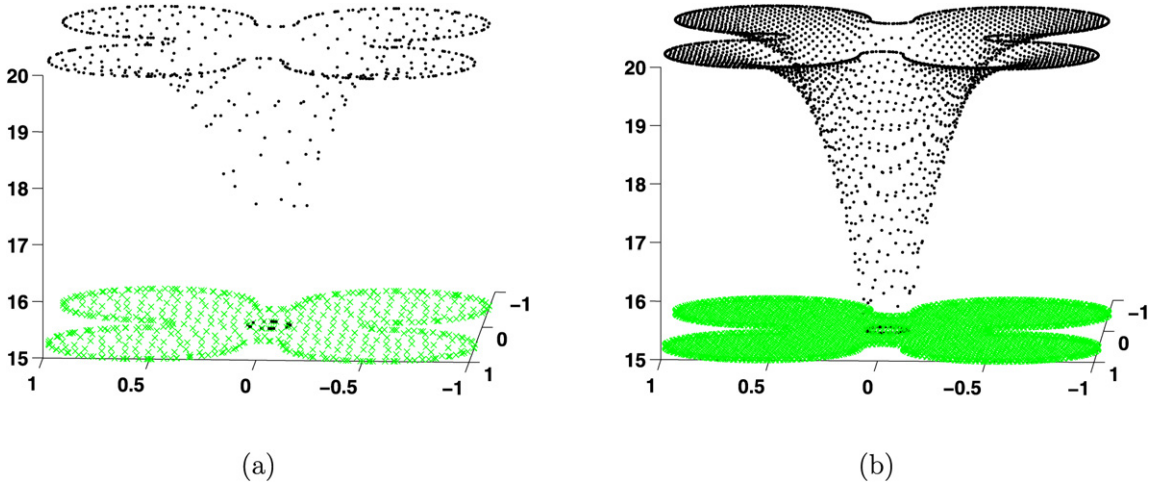

Fig. 7. 3D plots of the numerical solution for a method with second-order approximations of the derivatives. (a) Calculated on a mesh with 537 nodes, and (b) with 3198.

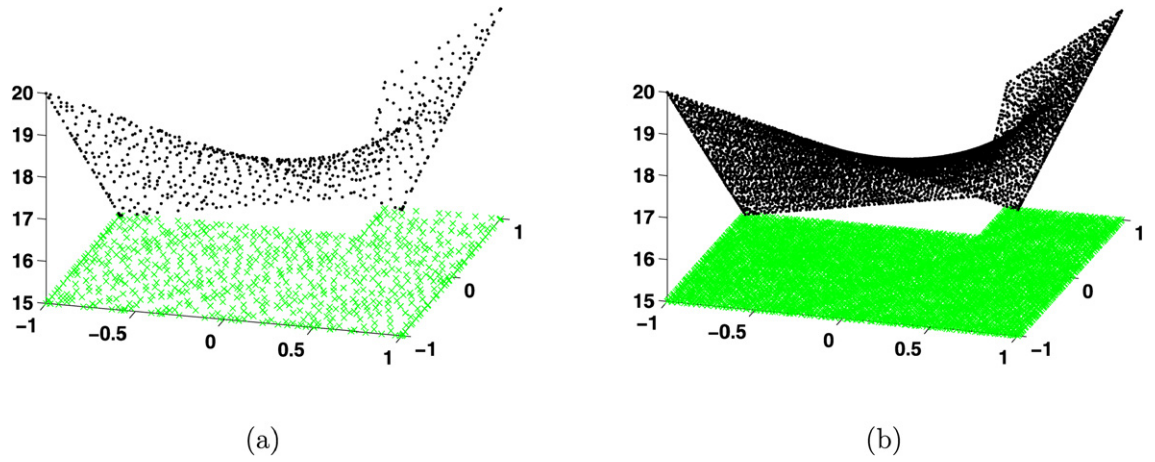

Fig. 8. 3D plots of the numerical solution for a method with second-order approximations of the derivatives. (a) Calculated on a mesh with 704 nodes, and (b) with 6754.

- A four-leaf clover whose boundary is defined by four circles of radius 0.45 located at positions $(0.55, 0.55)$, $(0.55, -0.55)$, $(-0.55, 0.55)$, and $(-0.55, -0.55)$, four circles of radius 0.10 at positions $(0.55, 0)$, $(-0.55, 0)$, $(0, 0.55)$ and $(0, -0.55)$, and an inner hole of radius 0.1 at the origin (see Fig. 6(a)).
- A polygon as shown in Fig. 6(b) obtained by cropping a rectangle of size 0.5 by 1.35 from a square of side 2 centered at the origin.

These domains have been meshed by randomly perturbing the slopes and spacing of two sets of parallel lines; the nodes have been located along the lines of one of the sets, at a small distance from the intersections with the lines in the other set. The resulting meshes, shown in Fig. 6, have a uniform node density, still having an irregular distribution.

Fig. 7 shows the numerical solution for problem (5) for the domain in Fig. 6(a), with $g(x, y) = 20$ in the outer boundary and $g(x, y) = 15$ in the inner one, with a mesh generated by using 537 (left plot) and 3198 (right plot) nodes. The observed gap in the neck of the numerical solution in the left plot is due to the lack of graphical interpolation between the values of the solution at the nodes. Fig. 8 shows the results for the domain in Fig. 6(b) with

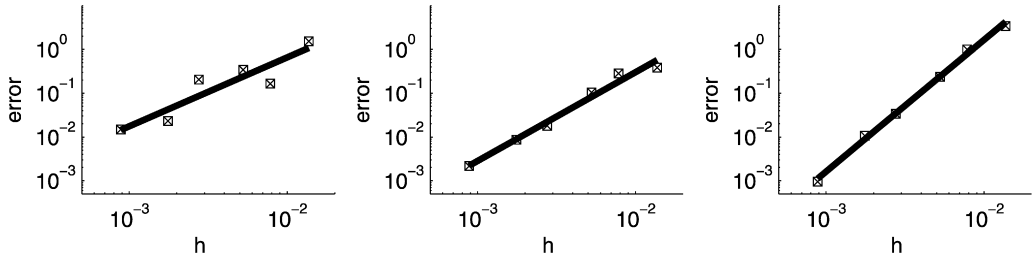

Fig. 9. Error of the numerical solution as a function of the mesh size h for the locally first-order (left plot), second-order (middle plot), and third-order (right plot) methods.

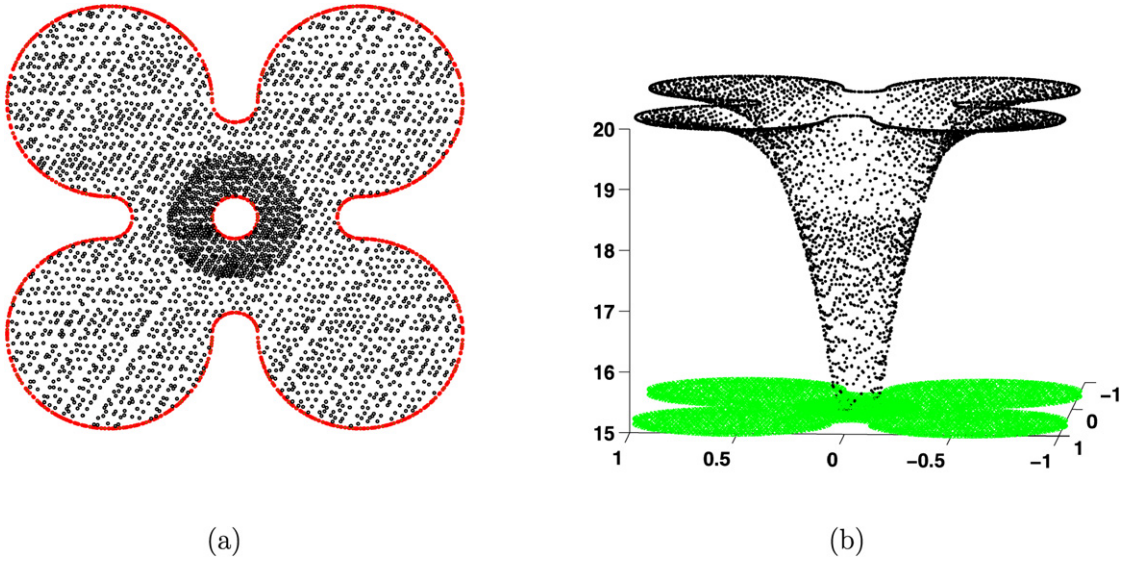

Fig. 10. Results obtained using a adapted variable density mesh (a) and the corresponding solution (b) obtained with our method.

boundary conditions obtained by linear interpolation of the values 20, 18, 20, 18, 16, and 15 assigned to the vertices of the domain, starting from the bottom left corner and turning anti-clockwise.

The plots in Figs. 7 and 8 show a smooth numerical solution, very accurate and without any spurious visible artifacts. Let us emphasize that, although Figs. 7 and 8 contain a small number of nodes, computations have also been performed with considerably high numbers (up to around 30,000). The number of nodes in the plots has been decided for clarity purposes, to show the absence of inconsistent values in the result.

One of the advantages of the generalized finite difference method developed in this paper is the possibility to obtain a numerical solution whose order of accuracy is constant throughout the solution. Thus, although the mesh be irregular, all the stars have exactly the same theoretical local order of approximation. Hence, a global order of accuracy of the method, independent of the mesh, could be expected when the node density increases.

Let us consider the numerical solution of problem (5) in a family of meshes with a decreasing density of nodes obtained by systematically removing nodes from a very fine mesh, generated as in Fig. 6. The position of the remaining nodes is perturbed again, except for a number of them, whose positions are fixed so that the error of the solution on each mesh can be determined. The numerical solution at the nodes which are common to all the meshes has been compared to the solution obtained on the finest mesh by means of the (locally) third-order accurate method, our best approximation to the exact solution of the problem.

Fig. 9 plots the Euclidean norm of the errors of the numerical solution at the common nodes as a function of mesh size, calculated as the mean of the star diameters, i.e. $h = (\sum h_q)/T$, where h_q is the star diameter of the star around node w_q and T is the total number of nodes in the mesh. Similar results have been obtained using both other norms and other definitions of h, such as the maximum star diameter in the mesh. Fig. 9 also shows the linear regression of the results. The locally first-, second-, and third-order accurate methods present slopes 1.574, 2.028, and 3.008, respectively, which approximate the theoretical order of accuracy. Besides, the linear correlation factor (R^2) increases with the order of accuracy. For a first order method R^2 is 0.842, for a second order 0.979, and for a third order 0.996. These results, which are representative of those obtained for a large set of numerical experiments, suggest that it is possible to determine a global order of accuracy for the method.

The good adaptivity properties of our method can be easily illustrated, since the number of nodes in any subregion of the domain can be easily enlarged locally with the additional cost of only recalculating the stencils in the nodes inside the refined region and correspondingly enlarging the stiffness matrix. The stencils for the nodes in the rest of the domain and the corresponding entries in the stiffness matrix remain untouched. In fact, the increase in cost due to adaptivity of the mesh is linear in the number of nodes added in the mesh. Fig. 10(a) shows the mesh in Fig. 6(a) with a large number of nodes around the inner circle, and Fig. 10(b) shows the result obtained with the method showing an increased accuracy in the neck of the solution.

7. Conclusions and future work

A mesh generation technique which allows the generation of finite difference stars with the properties of a Coatmèlec lattice in every node has been presented and applied to the development of generalized finite difference methods for partial differential equations. In order to illustrate the technique, an elliptic equation in two domains have been studied using several irreg-

ular meshes. The results show the good accuracy of the method when the mesh is refined. In fact, the constant local order of accuracy of the stars results in a unique global order of accuracy of the method. This may be considered one of the main advantages of these GFDMs.

An important problem with the mesh generation technique developed in this paper, which also applies to standard techniques, is that the resulting global matrix of the GFDM may be ill-conditioned when the density of the nodes in the mesh is very inhomogeneous, even when the local matrix for each star is well-conditioned in practice. The development of techniques for dealing with this problem is currently a very interesting open problem.

The results presented in this paper for elliptic equations with Dirichlet boundary conditions can be extended easily to other linear boundary conditions (Neumann or Robin). Moreover, the application of the method developed in this paper to evolution equations, both linear and nonlinear, require the analysis of the problems of stability and convergence, outside the scope of this paper, which constitutes a very interesting research topic for the future.

Coatmèlec lattices can be easily generalized to n dimensions by using a recursive definition [30]: the nodes are distributed in n-dimensional hyperplanes by means of using $(n-1)$-dimensional Coatmèlec lattices in each hyperplane. The number of nodes in the star is a binomial coefficient related to the dimension, the order of accuracy and the order of the derivative. The extension of our algorithm to three-dimensions is straightforward but requires an efficient algorithm to obtain the nodal connectivity in such a case, currently under development.

Acknowledgements

This work has been funded by projects FIS2005-03191, FIS2005-01189 and TIN2006-12890 from the Spanish Ministry of Education and Science and the project UV-AE-20070220 from the Universitat de València (Spain). This work has been partially supported by the Structural Funds of the European Regional Development Fund (ERDF). M.-A. G.-M. acknowledges useful discussions with G. Renversez and project HF2005-0172 from the Spanish Ministry of Education and Science. F.G. acknowledges project MTM 2004-06015-C02-01 from FEDER and DGI (Spain), project ACOMP07/088 from the Generalitat Valenciana (Spain), project PAID-06-08 from Universidad Politécnica de Valencia (Spain), and project DPI2008-02953, from Ministerio de Ciencia e Innovación (Spain).

References

[1] S.R. Idelsohn, E. Oñate, Comput. Meth. Appl. Mech. Eng. 195 (2006) 4681.
[2] R.A. Gingold, J.J. Monaghan, Mon. Not. Roy. Astron. Soc. 181 (1977) 375.
[3] T. Liszka, J. Orkisz, Comput. Struct. 11 (1980) 83.
[4] A.A. Tseng, S.X. Gu, Comput. Struct. 31 (1989) 319.
[5] P. Lancaster, K. Salkauskas, Math. Comput. 37 (1981) 141.
[6] B. Nayroles, G. Touzot, P. Villon, Comput. Mech. 10 (1992) 307.
[7] T. Belytschko, Y. Liu, L. Gu, Int. J. Numer. Meth. Eng. 37 (1994) 229.
[8] R. von Misses, Nat. Bur. Stand. Appl. Math. Ser. 18 (1952) 1.
[9] R.H. MacNeal, Q. Appl. Math. 11 (1953) 295.
[10] G.E. Forsythe, W.R. Wasow, Finite-Differential Methods for Partial Differential Equations, Wiley, New York, 1960.
[11] P.S. Jensen, Comput. Struct. 2 (1972) 17.
[12] N. Perrone, R. Kao, Comput. Struct. 5 (1975) 45.
[13] V. Girault, SIAM J. Numer. Anal. 11 (1974) 260.
[14] K.C. Chung, Numer. Heat Transfer 4 (1981) 345.
[15] J.J. Benito, F. Ureña, L. Gavete, Appl. Math. Model. 25 (2001) 1039.
[16] Y. Luo, U. Haussler-Combe, Comput. Meth. Appl. Mech. Eng. 191 (2002) 1421.
[17] L. Gavete, M.L. Gavete, J.J. Benito, Appl. Math. Model. 27 (2003) 831.
[18] K.C. Chung, T.H. Yao, SIAM J. Numer. Anal. 14 (1977) 735.
[19] X.Z. Liang, Acta Sci. Natur. Univ. Jilin 1 (1979) 27 (in Chinese).
[20] X.-Z. Liang, L.-H. Cui, J.-L. Zhang, J. Comput. Appl. Math. 163 (2004) 177.
[21] X.Z. Liang, Ch.M. Lu, in: L.L. Schumaker, C. Chui (Eds.), Approximation Theory IX, vol. 1: Theoretical Aspects, Vanderbilt University Press, Nashville, TN, 1998, p. 189.
[22] C. Coatmèlec, Ann. Sci. École Norm. Sup. 83 (1966) 271.
[23] C.K. Chui, M.J. Lai, in: B.L. Lin, S. Simon (Eds.), Nonlinear and Convex Analysis, Marcel Dekker, New York, 1987, p. 23.
[24] N. Wirth, Algorithms + Data Structures = Programs, Prentice-Hall, Englewood Cliffs, NJ, 1975.
[25] L. Guibas, J. Stolfi, ACM Trans. Graph. 4 (1985) 75.
[26] S. Fortune, Algorithmica 2 (1987) 153.
[27] P. Su, R.L.S. Drysdale, in: J. Snoeyink (Ed.), Proceedings of the Eleventh Annual Symposium on Computational Geometry, ACM Press, New York, 1995, p. 61.
[28] C.L. Lawson, in: J.R. Rice (Ed.), Mathematical Software III, Academic Press, New York, 1977, p. 161.
[29] M. Gasca, J.I. Maeztu, Numer. Math. 39 (1982) 1.
[30] M.A. García-March, F. Giménez, F.R. Villatoro, P. Fernández de Córdoba, Appl. Math. Lett. (2008), submitted for publication.