Modelling, Analysis and Simulation

A variational meshfree method for solving time-discrete diffusion equations

J.K. Krottje

REPORT MAS-E0319 DECEMBER 10, 2003

ISSN 1386-3703

A variational meshfree method for solving time-discrete diffusion equations

ABSTRACT
A meshfree method is developed for solving time-discrete diffusion equations that arise in models in brain research. Important criteria for a suitable method are flexibility with respect to domain geometry and the ability to work with very small moving sources requiring easy refinement possibilities. One part of the work concerns a meshfree discretization of the modified Helmholtz equation based on the related minimization problem and a local least squares function approximation. In a second part a node choosing algorithm is presented that moves around randomly distributed nodes for optimizing the node distribution and varying the node density as needed. The method is illustrated by two numerical tests.

2000 Mathematics Subject Classification: 35K57, 35J20, 65M99
Keywords and Phrases: parabolic equations, meshfree methods, variational methods, computational neurobiology
Note: Work carried out under project MAS1.3---Applications from the Life Sciences, continued in 2002 as MAS1.1---PDEs in the Life Sciences

A VARIATIONAL MESHFREE METHOD FOR SOLVING TIME-DISCRETE DIFFUSION EQUATIONS

J.K.Krottje
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

j.k.krottje@cwi.nl

Abstract.
A meshfree method is developed for solving time-discrete diffusion equations that arise in models in brain research. Important criteria for a suitable method are flexibility with respect to domain geometry and the ability to work with very small moving sources requiring easy refinement possibilities. One part of the work concerns a meshfree discretization of the modified Helmholtz equation based on the related minimization problem and a local least squares function approximation. In a second part a node choosing algorithm is presented that moves around randomly distributed nodes for optimizing the node distribution and varying the node density as needed. The method is illustrated by two numerical tests.

2000 Mathematics Subject Classification: 35K57, 35J20, 65M99.
Keywords and Phrases: diffusion equations, meshfree methods, variational methods, computational neurobiology.
Note: Work carried out under project MAS1.3—Applications from the Life Sciences, continued in 2002 as MAS1.1—PDEs in the Life Sciences.

1. Introduction

In the present paper a meshfree method is presented for solving time-discrete diffusion equations. This method is meant to be used for the simulation of certain models used in brain research. Such models describe mechanisms behind the development of the nervous system and in particular the formation of the connections between the nerve cells [10]. The resulting equations are constituted by two systems. One of the systems is a set of diffusion equations for certain chemicals (attractants and repellents) that is coupled to the other system which consists of nonlinear ODEs describing the growth of the connection forming structures, i.e., axons. The diffusion equations contain moving sources which are small compared to the domain and their strength and movement may depend on the solution of the ODEs. The nonlinear ODEs depend on the solutions of the diffusion equations, and gradients thereof, evaluated along solution paths in the space domain.

For the sake of clarity, we consider an example consisting of one diffusion equation and one ODE [13].

$$\frac{\partial}{\partial t}\rho(\mathbf{x},t) = (d\Delta - \kappa)\rho(\mathbf{x},t) + S(\mathbf{x}-\mathbf{r}_0) + S(\mathbf{x}-\mathbf{r}(t)),$$
$$\frac{d}{dt}\mathbf{r}(t) = \nabla\rho(\mathbf{r}(t),t),$$

where ρ is some concentration, S is a source profile with compact support, and $\mathbf{r}_0$ and $\mathbf{r}(t)$ are two source locations of which $\mathbf{r}(t)$ moves in the direction of higher concentrations of ρ. A first-order discretization in time of this model, where the ODE is handled explicitly and the diffusion equation implicitly, is

$$\begin{aligned}\Big(1-\delta t(d\Delta-\kappa)\Big)\rho^{n+1}(\mathbf{x}) &= \rho^n(\mathbf{x})+\delta t\Big(S(\mathbf{x}-\mathbf{r}_0)+S(\mathbf{x}-\mathbf{r}^{n+1})\Big),\\ \mathbf{r}^{n+1} &= \mathbf{r}^n+\delta t\Big(\nabla\rho^n(\mathbf{r}^n)\Big),\end{aligned}$$

where the superscripts n and $n+1$ denote different time levels and δt is the size of a time step. Clearly, we have to solve in every time step an elliptic equation and this step will comprise the majority of the work. Time discretizations of Runge-Kutta type will require the calculation of several of such equations per time step.

This article is focused on the numerical solution of these elliptic equations. The combined challenges of small, moving sources and flexible domain geometry suggests the use of a meshfree approach. The basic idea behind meshfree methods is to work with an arbitrary set of nodes instead of a grid. While this makes it easy to handle refinement and flexible domain geometries, function approximation and solvability of resulting systems become more complicated.

In most meshfree methods solving elliptic equations starts with the definition of an approximation space in which a best approximation of the solution is sought. This space is defined by selecting a set of basis functions, which in all cases forms a partition of unity to guarantee at least first-order approximation. While in finite element methods the basis functions are chosen with respect to a partition of the domain, meshfree methods form a basis by assigning functions with compact supports to each node of an arbitrary node set. Here the function's supports have to cover the whole domain, while the overlap has to be minimal so that the resulting linear equation systems become as sparse as possible. More on meshfree methods can be found in the overview articles [1, 7, 14].

In the method developed in this paper the approximation space is not defined through a set of basis functions but as the image of a linear map. This map assigns to every combination of function values on the nodes a piecewise smooth function on the domain by using a least squares approximation locally. In this way the function values on the nodes parametrize the approximation space that will be used in a Galerkin procedure. Because the approximating functions are piecewise multinomials the integrals in the resulting matrices can be evaluated exactly. This is in contrast to methods like DEM [16] and EFG [2] or variants thereof, where a quadrature rule is needed because of the use of moving least squares interpolants.

A Voronoi diagram [4, 6] based on the nodes is used for finding neighboring nodes and for glueing local approximations together to form a global approximation. Some other meshfree methods that use Voronoi diagrams (or the related Delaunay triangulations) are the Natural Element Method [17] and the Meshless Finite Element Method [12]. Both methods, however, use an approximation space constructed by choosing a set of basis functions.

For the construction of suitable node sets different techniques are available, see for example [15, 3]. Here an algorithm is presented that makes use of Voronoi diagrams and shifting nodes to 'regularize' node sets.

The contents of the paper is as follows. We start in Section 2 with meshfree function approximation based on a local least squares approximation. This is followed by a description of the discretization of the equation in Section 3. In Section 4 more practical considerations concerning the computation of the discretization are discussed. A way for dealing with different domains and refinement in a meshfree context is examined in Section 5, followed by Section 6 with a numerical test example. Finally, Section 7 summarizes the paper.

2. Meshfree function approximation

This section deals with the meshfree function approximation used in the discretization of the equation. We will start with a description of local least squares approximation, examine its convergence and use it for a global approximation.

2.1. **Local Least Squares Approximation.** Given a function $f\colon \mathbb{R}^2 \to \mathbb{R}$ and some node set $\{\mathbf{x}_0 + h\mathbf{x}_1, \ldots, \mathbf{x}_0 + h\mathbf{x}_n\}$, we want to approximate the function in the disc with center $\mathbf{x}_0$ and radius $h > 0$, using the values $\{f(\mathbf{x}_0 + h\mathbf{x}_1), \ldots, f(\mathbf{x}_0 + h\mathbf{x}_n)\}$. We choose the approximant to be a linear combination of multinomials which are maximal of second order, i.e.,

$$p^h(\mathbf{x}) = \boldsymbol{\alpha} \cdot \mathbf{b}(\mathbf{x}) \quad \text{with} \quad \mathbf{b}(\mathbf{x}) = \begin{pmatrix} 1 & x & y & x^2 & xy & y^2 \end{pmatrix}^T, \tag{1}$$

where $\mathbf{x} = (x, y)$ and $\boldsymbol{\alpha} \in \mathbb{R}^6$ is determined by minimizing

$$\sum_{i=1}^{n} \left| p^h(\mathbf{x}_0 + h\mathbf{x}_i) - f(\mathbf{x}_0 + h\mathbf{x}_i) \right|^2. \tag{2}$$

From now on we will assume that $\mathbf{x}_0 = \mathbf{0}$. A particular choice of $\mathbf{x}_0$ will not influence any approximation properties, because a translation in the domain of the least squares problem can be viewed as a linear change of basis functions, meaning that the approximation is in the same function space.

If we define

$$B(h) = \begin{pmatrix} \mathbf{b}(h\mathbf{x}_1) \Big| \ldots \Big| \mathbf{b}(h\mathbf{x}_n) \end{pmatrix} \quad \text{and} \quad \mathbf{F}(h) = \begin{pmatrix} f(h\mathbf{x}_1) \\ \vdots \\ f(h\mathbf{x}_n) \end{pmatrix},$$

substitution of (1) into (2), will result in

$$\boldsymbol{\alpha}^T \big(B(h)B(h)^T\big)\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T B(h)\mathbf{F}(h) + \mathbf{F}(h)^T\mathbf{F}(h), \tag{3}$$

which has to be minimized over $\boldsymbol{\alpha}$. The 6×6 matrix $B(h)B(h)^T$ is positive semi-definite and positive definite if $\det\big(B(h)B(h)^T\big) \neq 0$. In the latter case a unique minimum exists that is determined by

$$\big(B(h)B(h)^T\big)\boldsymbol{\alpha} = B(h)\mathbf{F}(h). \tag{4}$$

The entries of the matrix $B(h)B(h)^T$ can be written as

$$(B(h)B(h)^T)_{i,j} = \sum_{k=1}^{n} b_i(h\mathbf{x}_k) b_j(h\mathbf{x}_k). \tag{5}$$

Ill-conditioning. For finding the local approximation we have to solve equation (4). It turns out that for small h the matrix $B(h)B(h)^T$ is ill-conditioned for all sets $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$. To obtain a rough lower bound for the condition number we can use the fact that the diagonal elements of a symmetric matrix are bounded by the smallest and largest eigenvalue, which can be easily deduced using the *interlacing eigenvalues theorem for bordered matrices* [11].

The $(1,1)$-entry of $B(h)B(h)^T$ is equal to $\sum_{i=1}^n 1 = n$, while we also have, by using (5) and considering the diagonal elements $(B(h)B(h)^T)_{i,i}$, $i = 4, 5, 6$,

$$4\lambda_{\min} \leq h^4 \sum_{j=1}^n |\mathbf{x}_j|^4 \leq nh^4 \max_j |\mathbf{x}_j|^4 \quad \Longrightarrow \quad \frac{4}{h^4 \max_j |\mathbf{x}_j|^4} \leq \frac{\lambda_{\max}}{\lambda_{min}}. \tag{6}$$

Therefore $\operatorname{cond}(B(h)B(h)^T) \geq \mathcal{O}\left(h^{-4}\right)$ and direct calculation of the approximation could be an error-prone procedure. A more stable way of calculating the approximation is to use scaled basis functions $\hat{b}_i(\mathbf{x}) = b_i(\frac{1}{h}\mathbf{x})$, $i = 1, \dots, 6$. Then the matrices in equation (4) become independent of h and the ill-conditioning for small h will disappear.

2.2. **Convergence Properties.** To examine the convergence order of such an approximation, we will assume from now on that all points $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ are situated in the unit circle, that at least for one point $\|\mathbf{x}_i\|_2 = 1$, and that the resulting matrix $B(1)B(1)^T$ is invertible. An example configuration of points is shown in Figure 1. For an arbitrary point $\mathbf{x}$ in the unit circle we will examine now $|p^h(h\mathbf{x}) - f(h\mathbf{x})|$.

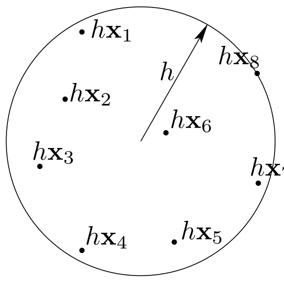

FIGURE 1. Example configuration of points and the related circle.

First we define $S(h) = \operatorname{diag}\left(1, h, h, h^2, h^2, h^2\right)$ and $B = B(1)$, so that we can write $\mathbf{b}(h\mathbf{x}) = S(h)\mathbf{b}(\mathbf{x})$ and $B(h) = S(h)B$. Using this we have from (4)

$$S(h)\left(BB^T\right)S(h)\boldsymbol{\alpha}(h) = S(h)B\mathbf{F}(h) \implies S(h)\boldsymbol{\alpha}(h) = \left(BB^T\right)^{-1}B\mathbf{F}(h)$$

and the approximation $p^h(h\mathbf{x})$ can be written as

$$p^h(h\mathbf{x}) = \boldsymbol{\alpha}(h) \cdot \mathbf{b}(h\mathbf{x}) = \mathbf{b}(\mathbf{x})^T S(h)\boldsymbol{\alpha}(h) = \mathbf{b}(\mathbf{x})^T \left(BB^T\right)^{-1} B\mathbf{F}(h).$$

For the difference $p^h(h\mathbf{x}) - f(h\mathbf{x})$ a Taylor series expansion can be written down by making Taylor series of $\mathbf{F}(h)$ and $f(h\mathbf{x})$. It turns out that the lower order coefficients cancel out due to the following equalities,

$$\begin{gathered}
\mathbf{v}^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 1, \qquad \mathbf{v}^T \begin{pmatrix} Df(\mathbf{0})(\mathbf{x}_1) \\ \vdots \\ Df(\mathbf{0})(\mathbf{x}_n) \end{pmatrix} = Df(\mathbf{0})(\mathbf{x}), \\
\mathbf{v}^T \begin{pmatrix} D^2 f(\mathbf{0})(\mathbf{x}_1, \mathbf{x}_1) \\ \vdots \\ D^2 f(\mathbf{0})(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} = D^2 f(\mathbf{0})(\mathbf{x}, \mathbf{x}),
\end{gathered} \tag{7}$$

where $\mathbf{v}$ is defined by $\mathbf{v}^T \equiv \mathbf{b}(\mathbf{x})^T (BB^T)^{-1} B$. Here we used the m-th order Fréchet derivative $D^m(f)(\mathbf{0})\colon (\mathbb{R}^2)^m \to \mathbb{R}$ of f at $\mathbf{0}$ which is a multi-linear operator and defined by

$$D^m f(\mathbf{0})(\boldsymbol{\eta}^1, \dots, \boldsymbol{\eta}^m) = \left(\prod_{i=1}^m (\boldsymbol{\eta}_1^i \partial_1 + \boldsymbol{\eta}_2^i \partial_2) \right) f(\mathbf{z}) \Bigg|_{\mathbf{z}=\mathbf{0}} .$$

Each of these equations can be associated with a least squares approximation problem that has an exact solution because the approximated function is a low order multinomial.

It follows that

$$p^h(h\mathbf{x}) = f(h\mathbf{x}) + \tfrac{1}{6} C h^3 + \mathcal{O}\left(h^4\right)$$

with

$$C = \mathbf{b}(\mathbf{x})^T \left(BB^T\right)^{-1} B \begin{pmatrix} D^3 f(\mathbf{0})(\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_1) \\ \vdots \\ D^3 f(\mathbf{0})(\mathbf{x}_n, \mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} - D^3 f(\mathbf{0})(\mathbf{x}, \mathbf{x}, \mathbf{x}). \tag{8}$$

For arbitrary $\mathbf{z} \in \mathbb{R}^2$ with $|\mathbf{z}| \leq 1$, we have $\|\mathbf{b}(\mathbf{z})\|_2 \leq \sqrt{6} \|\mathbf{b}(\mathbf{z})\|_\infty \leq \sqrt{6}$ and

$$\begin{aligned}
D^3 f(\mathbf{0})(\mathbf{z}, \mathbf{z}, \mathbf{z}) &= (z_1 \partial_1 + z_2 \partial_2)^3 f(z_1, z_2) \Big|_{z_1=0, z_2=0} \\
&\leq 8 \max_{i,j,k=1,2} |\partial_i \partial_j \partial_k f(\mathbf{0})|,
\end{aligned}$$

which, because $|\mathbf{x}|, |\mathbf{x}_1|, \dots, |\mathbf{x}_n| \leq 1$, yield together

$$|C| \leq (48n \|(BB^T)^{-1}\|_\infty + 8) \max_{i,j,k=1,2} |\partial_i \partial_j \partial_k f(\mathbf{0})|.$$

For a real square $m \times m$-matrix A we have the inequality of Hadamard [18] which states that $|\det(A)| \leq (m+1)^{(m+1)/2} / 2^m$. Applying the general expression for matrix inverses using cofactors on BB^T gives

$$((BB^T)^{-1})_{i,j} = \frac{1}{\det(BB^T)} (-1)^{i+j} \det(BB^T[j,i]),$$

where $BB^T[i,j]$ is a matrix BB^T with row i and column j deleted. One can now estimate

$$\|(BB^T)^{-1}\|_\infty \leq \frac{81}{2n \det(\frac{1}{n} BB^T)},$$

which by denoting $D(0;h) = \{|\mathbf{x}| \leq 1\}$, results finally in

$$\sup_{\mathbf{x} \in D(0;h)} |p^h(\mathbf{x}) - f(\mathbf{x})| \leq \left(\frac{324}{\det(\frac{1}{n} BB^T)} + \frac{4}{3} \right) \max_{i,j,k=1,2} |\partial_i \partial_j \partial_k f(\mathbf{0})| \, h^3, \tag{9}$$

for h small enough.

Therefore we can state that for node sets in the domain of which the subsets used for local approximation can be enclosed in circles of radius h and for which all $\det(\frac{1}{n}BB^T)$ are bounded from below by some constant, the approximation is of third order. These used subsets we will call the local node sets.

On the other hand, for arbitrary local node sets, $\det(\frac{1}{n}BB^T)$ can be arbitrary close to zero, making the third order constant larger, possibly resulting in a bad approximation. Because the idea of meshfree methods is to start with arbitrary node sets, we will after choosing subsets of the global node set, test their approximation ability by evaluating $\det(\frac{1}{n}BB^T)$. For determinant values too small we will add more points from the global node set, repeating this procedure until the determinant is above the required constant.

Derivative approximation. For the approximation of derivatives we expect similar behavior with one order lower accuracy, i.e., second order. To examine how well the derivatives are approximated, we define the matrices $D_1, D_2 \in \mathbb{R}^{6\times 6}$ by

$$D_1 = \begin{pmatrix} 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&2&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&0 \end{pmatrix} \quad \text{and} \quad D_2 = \begin{pmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&2&0&0&0 \end{pmatrix}.$$

These represent the actions of the partial derivatives with respect to the basis functions $b_1, \ldots, b_6$ in such a way that for functions $g\colon \mathbb{R}^2 \to \mathbb{R}$ that are arbitrary linear combinations of the basis functions $g(\mathbf{x}) = \boldsymbol{\alpha} \cdot \mathbf{b}(\mathbf{x})$ we have

$$\partial_i g(\mathbf{x}) = \boldsymbol{\alpha} \cdot D_i \mathbf{b}(\mathbf{x}) \qquad (i = 1, 2).$$

With respect to the matrix $S(h)$ these matrices obey $D_i S(h) = \frac{1}{h} S(h) D_i$. This is a direct consequence of the fact that for arbitrary functions $f\colon \mathbb{R} \to \mathbb{R}$ with $f(hx) = h^p f(x)$ for some constant p, it follows that $Df(hx) = h^{p-1} Df(x)$.

For the approximation of the i-th partial derivative we can write now

$$\partial_i p^h(h\mathbf{x}) = \boldsymbol{\alpha}(h) \cdot D_i \mathbf{b}(h\mathbf{x}) = D_i^T \mathbf{b}(\mathbf{x})^T \tfrac{1}{h} S(h) \boldsymbol{\alpha}(h) = D_i^T \mathbf{b}(\mathbf{x})^T \left(BB^T\right)^{-1} B \tfrac{1}{h} \mathbf{F}(h).$$

Differentiating the equations (7) gives

$$D_i^T v^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \frac{\partial}{\partial x_i} 1 = 0, \quad D_i^T v^T \begin{pmatrix} Df(\mathbf{0})(\mathbf{x}_1) \\ \vdots \\ Df(\mathbf{0})(\mathbf{x}_n) \end{pmatrix} = \frac{\partial}{\partial x_i} Df(\mathbf{0})(\mathbf{x}) = \partial_i f(\mathbf{0}),$$

$$D_i^T v^T \begin{pmatrix} D^2 f(\mathbf{0})(\mathbf{x}_1, \mathbf{x}_1) \\ \vdots \\ D^2 f(\mathbf{0})(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} = \frac{\partial}{\partial x_i} D^2 f(\mathbf{0})(\mathbf{x}, \mathbf{x}) = 2D(\partial_i f)(\mathbf{0})(\mathbf{x}),$$

which again leads to vanishing coefficients in the Taylor series expansion of $\partial_i p^h(h\mathbf{x}) - \partial_i f(h\mathbf{x})$, resulting in

$$\partial_i p^h(h\mathbf{x}) = \partial_i f(h\mathbf{x}) + \tfrac{1}{6} C' h^2 + \mathcal{O}\left(h^3\right)$$

with

$$C_i' = D_i^T \mathbf{b}(\mathbf{x})^T \left(BB^T\right)^{-1} B \begin{pmatrix} D^3 f(\mathbf{0})(\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_1) \\ \vdots \\ D^3 f(\mathbf{0})(\mathbf{x}_n, \mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} - D^3 f(\mathbf{0})(\mathbf{x}, \mathbf{x}, \mathbf{x}). \qquad (10)$$

A similar estimation as in the non-derivative case yields for h small enough

$$\sup_{\mathbf{x} \in D(0;h)} |\partial_i p^h(\mathbf{x}) - \partial_i f(\mathbf{x})| \leq \left(\frac{648}{\det(\frac{1}{n} BB^T)} + \frac{4}{3} \right) \max_{i,j,k=1,2} |\partial_i \partial_j \partial_k f(\mathbf{0})| \, h^2. \qquad (11)$$

2.3. **Quality of local node sets.** Above it was stated that we use $\det(\frac{1}{n}BB^T)$ of a local node set as a measure of the approximation quality. In this paragraph we want to investigate this further.

In the calculation of the determinant a circle of radius h was used which contains all nodes of the local node set with the restriction that at least one of the nodes is on the circle. The convergence results (9) and (11) then hold for arbitrary points in the circle. In most cases the area in which we use the approximation is actually much smaller than the circle (e.g. a Voronoi tile containing the central node, See Figure 2) and probably somewhere in the middle of it. This means that there are many possibilities for choosing such a circle, as is illustrated in Figure 2. Here, a local node set is shown together with a polygonal area in which we are interested. The three circles, having radii 0.97 (solid circle), 1.00 and 1.10, respectively, are all suitable choices.

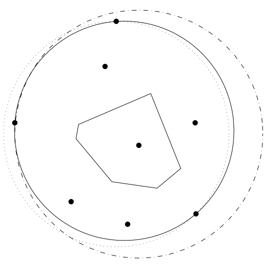

FIGURE 2. Three possible choices for the circle

Because the approximation is independent of the choice of the circle, we want the quality measure of the local node set to be also independent of this choice, which means that we have to make a particular choice. Before showing what a good choice is we will examine how $\det(\frac{1}{n}BB^T)$ responds to translations, rotations and scaling of the local node set. For this we introduce a translation vector $\mathbf{v} \in \mathbb{R}^2$, an orthogonal rotation matrix $Q \in \mathbb{R}^{2\times 2}$ and some real scaling constant $r > 0$. If we compare now $\det(\frac{1}{n}BB^T)$ for a node set $\{\mathbf{x}_i\}$, $i = 1, \ldots, n$ with $\det(\frac{1}{n}\tilde{B}\tilde{B}^T)$, where $\tilde{B}$ is made out of vectors $\mathbf{b}(rQ\mathbf{x}_i + \mathbf{v})$ instead of vectors $\mathbf{b}(\mathbf{x}_i)$ as with B, then it can be shown that

$$\det(\tfrac{1}{n}\tilde{B}\tilde{B}^T) = r^{16} \det(\tfrac{1}{n}BB^T).$$

This means that the measurement is invariant with respect to rotation and translation, but that the radius of the chosen circle strongly affects the value of the determinant. For example, choosing a circle which is twice as large, means that the distances between the nodes in the local node set measured relative to the circle radius h will be twice as small, yielding $r = 0.5$. This will result in a determinant which is $(0.5)^{16} = 1.5 \cdot 10^{-5}$ times as large. The determinants in Figure 2 are in ratio $1 : 0.61 : 0.13$.

A good circle choice is the smallest enclosing circle of the given local node set, so that the determinant is maximal. From now on, we will use this choice and assume that the area in which we will use our approximation is inside the smallest enclosing circle. Actually, the solid circle in Figure 2 is the smallest enclosing circle.

An algorithm for calculation of the smallest enclosing circle is described in [4], which is a randomized incremental algorithm with an expected time complexity of linear order in the number of nodes in the local node set and hence is very efficient.

2.4. **Global approximation.** In the previous subsections a method for local function approximation is discussed. Here we will use it for making a global approximation of a function, given an arbitrary set of nodes in our global domain. The basic idea is to divide the domain in disjoint sub-domains in which we use then the local approximation.

Such a division can be made out of our set of nodes by calculating the related Voronoi diagram [4]. In such a diagram every node has its own Voronoi tile, which consists of all points of the domain which are closer to the node associated with the tile than to every other node of the node set. This will make all the tiles disjoint and their union equal to $\mathbb{R}^2$ except for a set of which the points are equally close to two or more nodes. This set is called the Voronoi diagram. In the left picture of Figure 3 the Voronoi diagram of a particular node set is shown. We will use the unbounded Voronoi diagram to make a division for our bounded domain by connecting all nodes which are on the boundary of the domain by straight lines. This gives us an approximation of our domain which is of second order (with respect to integrals) in the distance between the nodes on the boundaries in case the boundaries are curved. The right picture of Figure 3 shows the division of the domain.

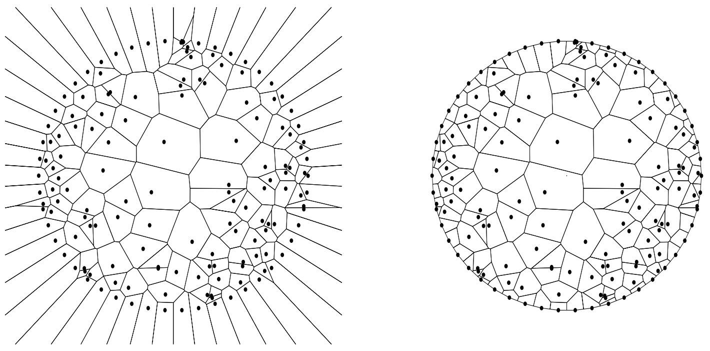

FIGURE 3. Voronoi diagram of the node set and the domain decomposition based on it.

Now for all the Voronoi tiles we have a local approximation giving us a global approximation in our domain except for the Voronoi diagram, which is of measure zero and therefore irrelevant, because we will use the global approximation for evaluating integrals.

Finding neighboring nodes. For finding the local approximation in a Voronoi tile a local set of nodes has to be found. Using the diagram, we can define nodes to be neighbors if and only if their Voronoi tiles have a common Voronoi edge of the diagram. In this way we can find a ring of neighbors of a node and also a second ring of neighbors of neighbors of a node. Proceeding until we have at least the number of neighbors we need, which is minimally six nodes in case of six basis functions, and the determinant $\det(\frac{1}{n}BB^T)$ is above some threshold, will result in the local node set.

Global approximation mapping. We are now ready to define a mapping which will map a function defined on the set of nodes $\mathcal{N}$ into the space $L^1(\Omega)$, where Ω is our domain. This mapping will be used in the subsequent section on the discretization of the PDE. We define the mapping $F_{\mathcal{N}}\colon \mathbb{R}^N \to L^1(\Omega)$ such that for a given node function $\mathbf{f} \in \mathbb{R}^N$, a tile Ω_i and its local approximation p_{Ω_i}, we have $F_{\mathcal{N}}(\mathbf{f}) = p_{\Omega_i}$ in tile Ω_i. This defines $F_{\mathcal{N}}(\mathbf{f})$ in every tile of the domain. What is left is the Voronoi diagram, which is a set with measure zero and because functions that differ on a set of zero measure are identical in L^1, the definition is complete.

If we define also a restriction map $G_{\mathcal{N}}\colon C^1(\Omega) \to \mathbb{R}^N$ by $G_{\mathcal{N}}(u) = u|_{\mathcal{N}}$ (pointwise restriction), we can write for an arbitrary function $u \in C^3(\Omega)$, and h defined as the maximal radius of all circles used in the local approximation,

$$\|u - F_{\mathcal{N}} \circ G_{\mathcal{N}}(u)\|_\infty \leq Kh^3, \tag{12}$$

for h small enough and where K is some positive constant. In a similar way we have for $i = 1, 2$,

$$\|\partial_i u - \partial_i(F_{\mathcal{N}} \circ G_{\mathcal{N}}(u))\|_\infty \leq K'h^2. \tag{13}$$

3. Discretization of the PDE

To discretize the PDE problem we will formulate it as a minimization problem and use the approximation technique from the previous section to end up with a discrete minimization problem. We then consider the solvability of the linear system that has to be solved for finding the minimum.

3.1. Minimization problem.

We will consider the elliptic problem

$$\begin{aligned} (d\Delta - \kappa)u + f &= 0, \quad \mathbf{x} \in \Omega, \\ \nabla u \cdot \mathbf{n} &= 0, \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{14}$$

with $d, \kappa > 0$ and $f \in L^2(\Omega)$. This boundary value problem can be written as the variational problem

$$K[u] = \min_{w \in H^1} K[w], \quad \text{with} \begin{cases} K[w] = A(w,w) - L(f,w), \\ A(v,w) = \int_\Omega \frac{1}{2} d\nabla v \cdot \nabla w + \frac{1}{2}\kappa v w \, dx, \\ L(f,w) = \int_\Omega f w \, dx, \end{cases} \tag{15}$$

which can be found in textbooks on elliptic PDEs, for example [8].

To find a numerical solution of the minimization problem we will calculate first an approximation of the integral for a given node function $\mathbf{w} \in \mathbb{R}^N$. We do this by plugging $F_{\mathcal{N}}[\mathbf{w}]$ and $F_{\mathcal{N}}[\mathbf{f}]$ into integrals $A(w,w)$ and $L(f,w)$ of (15), where we define $\mathbf{f} = G_{\mathcal{N}}(f)$, yielding

$$\begin{aligned} A(F[\mathbf{w}],F[\mathbf{w}]) &= \int_\Omega \tfrac{1}{2} d\,\nabla(F[\mathbf{w}])\cdot\nabla(F[\mathbf{w}]) + \tfrac{1}{2}\kappa\,F[\mathbf{w}]^2\,d\mathbf{x}, \\ L(F[\mathbf{f}],F[\mathbf{w}]) &= \int_\Omega F[\mathbf{f}]F[\mathbf{w}]\,d\mathbf{x}, \end{aligned} \tag{16}$$

where we have dropped the subscript $\mathcal{N}$ for convenience. In order to write

$$\frac{1}{2}\mathbf{w}^T\hat{A}\mathbf{w} = A(F[\mathbf{w}],F[\mathbf{w}]) \quad \text{and} \quad \mathbf{f}^T\hat{L}\mathbf{w} = L(F[\mathbf{f}],F[\mathbf{w}]),$$

for certain matrices $\hat{A}$ and $\hat{L}$, we define linear operators $P_i\colon \mathbb{R}^N \to \mathbb{R}^{n_i}$, such that for an arbitrary node function $\mathbf{w}\in\mathbb{R}^N$, $P_i\mathbf{w}\in\mathbb{R}^{n_i}$ equals the node function restricted to $\mathcal{N}_i$ (with the ordering inherited from $\mathcal{N}$).

If we denote $\mathcal{N}_i = \{\mathbf{x}_1^i,\dots,\mathbf{x}_{n_i}^i\}$, and define the matrix $B_i = (\mathbf{b}(\mathbf{x}_1^i)|\dots|\mathbf{b}(\mathbf{x}_{n_i}^i))$, we can write the least squares approximation $p_{\Omega_i}[\mathbf{w}]$ on Ω_i as

$$p_{\Omega_i}[\mathbf{w}](\mathbf{x}) = \left[(B_iB_i^T)^{-1}B_iP_i\mathbf{w}\right]\cdot\mathbf{b}(\mathbf{x}). \tag{17}$$

Let us write $\tilde{B}_i = (B_iB_i^T)^{-1}B_i$, so that we have $p_{\Omega_i}[\mathbf{w}](\mathbf{x}) = \tilde{B}_iP_i\mathbf{w}\cdot\mathbf{b}(\mathbf{x})$. Taking the gradient of such a function will result in an expression like

$$\nabla(p_{\Omega_i}[\mathbf{w}])(\mathbf{x}) = \begin{pmatrix} D_1^T\tilde{B}_iP_i\mathbf{w}\cdot\mathbf{b}(\mathbf{x}) \\ D_2^T\tilde{B}_iP_i\mathbf{w}\cdot\mathbf{b}(\mathbf{x}) \end{pmatrix},$$

where $D_{1,2}$ are the 6×6-matrices defined in Section 2, yielding

$$\begin{aligned} A(F[\mathbf{w}],F[\mathbf{w}]) &= \sum_i \int_{\Omega_i} \tfrac{1}{2}d\|\nabla(p_{\Omega_i}[\mathbf{w}])\|^2 + \tfrac{1}{2}\kappa(p_{\Omega_i}[\mathbf{w}])^2\,d\mathbf{x} \\ &= \sum_i \int_{\Omega_i} \Big\{ \tfrac{1}{2}d\,\mathbf{w}^TP_i^T\tilde{B}_i^T\Big(D_1\mathbf{b}(\mathbf{x})\mathbf{b}(\mathbf{x})^TD_1^T + D_2\mathbf{b}(\mathbf{x})\mathbf{b}(\mathbf{x})^TD_2^T\Big)\tilde{B}_iP_i\mathbf{w} \\ &\qquad + \tfrac{1}{2}\kappa\,\mathbf{w}^TP_i^T\tilde{B}_i^T\mathbf{b}(\mathbf{x})\mathbf{b}(\mathbf{x})^T\tilde{B}_iP_i\mathbf{w}\Big\}\,d\mathbf{x}. \end{aligned}$$

By defining $I_{\Omega_i} = \int_{\Omega_i}\mathbf{b}(\mathbf{x})\mathbf{b}(\mathbf{x})^T\,d\mathbf{x}$, this can be written as

$$\begin{aligned} &A(F[\mathbf{w}],F[\mathbf{w}]) \\ &\quad = \tfrac{1}{2}\mathbf{w}^T\left(\sum_{i=1}^N P_i^T\tilde{B}_i^T\big(d\,(D_1I_{\Omega_i}D_1^T + D_2I_{\Omega_i}D_2^T) + \kappa I_{\Omega_i}\big)\tilde{B}_iP_i\right)\mathbf{w} \\ &\quad = \tfrac{1}{2}\mathbf{w}^T\left(\sum_{i=1}^N P_i^TM_i^AP_i\right)\mathbf{w}, \end{aligned} \tag{18}$$

where we have written $M_i^A = \tilde{B}_i^T\big(d\,(D_1I_{\Omega_i}D_1^T + D_2I_{\Omega_i}D_2^T) + \kappa I_{\Omega_i}\big)\tilde{B}_i$, which is an $n_i\times n_i$-matrix. Therefore $\hat{A} = \sum_{i=1}^N P_i^TM_i^AP_i$. The $N\times N$-matrix $P_i^TM_i^AP_i$ is a large sparse matrix with the entries of matrix M_i^A put at locations dependent on the location of the nodes $\mathcal{N}_i$ in the ordering of $\mathcal{N}$.

In a similar way we have that

$$L(F[\mathbf{f}],F[\mathbf{w}]) = \mathbf{f}^T\left(\sum_{i=1}^N \tilde{P}_i^TB_i^TI_{\Omega_i}\tilde{B}_iP_i\right)\mathbf{w} = \mathbf{f}^T\left(\sum_{i=1}^N \tilde{P}_i^TM_i^LP_i\right)\mathbf{w}, \tag{19}$$

implying $\hat{L} = \sum_{i=1}^{N} \tilde{B}_i^T I_{\Omega_i} \tilde{B}_i$ so that $\hat{L}$ has the same sparsity structure as $\hat{A}$. Further, I_{Ω_i} is a symmetric matrix for every node i and therefore $\hat{A}$ and $\hat{L}$ are also symmetric. Our continuous minimization problem (15) has now been translated into a discrete minimization problem

$$\hat{K}[\mathbf{u}] = \min_{\mathbf{w}\in\mathbb{R}^N} \hat{K}[\mathbf{w}], \quad \text{with} \quad \hat{K}[\mathbf{w}] = \tfrac{1}{2}\mathbf{w}^T\hat{A}\mathbf{w} - \mathbf{f}^T\hat{L}\mathbf{w}. \tag{20}$$

3.2. **Solving the discrete problem.** If we assume that $\hat{A}$ is invertible, then we can rewrite $\hat{K}[\mathbf{w}]$ as

$$\hat{K}[\mathbf{w}] = \tfrac{1}{2}\big(\mathbf{w} - \hat{A}^{-1}\hat{L}\mathbf{f}\big)^T \hat{A}\big(\mathbf{w} - \hat{A}^{-1}\hat{L}\mathbf{f}\big) - \mathbf{f}^T\hat{L}^2\mathbf{f}. \tag{21}$$

The matrix $\hat{A}$ is positive semi-definite because $\mathbf{v}^T\hat{A}\mathbf{v} = 2A(F[\mathbf{v}], F[\mathbf{v}]) \geq 0$. Being also invertible would give positive-definiteness, which means that $\hat{K}[\mathbf{w}]$ is minimal for $\mathbf{w} = \mathbf{u}$, with $\mathbf{u} = \hat{A}^{-1}\hat{L}\mathbf{f}$ and $\hat{K}[\mathbf{u}] = -\mathbf{f}^T\hat{L}^2\mathbf{f}$.

Invertibility of $\hat{A}$. If the linear function $F\colon \mathbb{R}^N \to L^1(\Omega)$ is injective, meaning that $\dim(\mathrm{Im}(F)) = N$, then $\hat{A}$ must be invertible. If so we can define a norm on $\mathbb{R}^N$ by $\|\mathbf{v}\| \equiv \|F[\mathbf{v}]\|_{L^1}$. In finite dimension this norm is equivalent to the standard norm $\|\cdot\|_2$, meaning that there must be a constant $C_{\mathcal{N}} > 0$, such that for arbitrary $\mathbf{v} \in \mathbb{R}^N$, $\|F[\mathbf{v}]\|_{L^1} \geq C_{\mathcal{N}}\|\mathbf{v}\|_2$. This yields

$$\begin{aligned}
\mathbf{v}^T\hat{A}\mathbf{v} &= 2A(F[\mathbf{v}], F[\mathbf{v}]) = 2\sum_{i=1}^{N}\int_{\Omega_i} d|\nabla F[\mathbf{v}]|^2 + \kappa F[\mathbf{v}]^2\,dx \\
&\geq 2\kappa\sum_{i=1}^{N}\int_{\Omega_i} F[\mathbf{v}]^2\,dx \geq 2\kappa\int_{\Omega} dx\|F[\mathbf{v}]\|_{L^1}^2 \geq \big(2\kappa C_{\mathcal{N}}^2\int_{\Omega} dx\big)\mathbf{v}^T\mathbf{v}.
\end{aligned}$$

Therefore all eigenvalues of $\hat{A}$ are greater than zero and $\hat{A}$ is invertible.

In this paper we will take a practical approach and assume from now on that F is injective (and do not examine under which conditions this is true.) If, given a certain set of nodes, the choice of the local nodes sets is such that $\mathcal{N}_i$ consists of precisely 6 points and $\det(\frac{1}{n}B_iB^T) \neq 0$, than every local approximation multinomial is the unique interpolation multinomial of the six nodes and their function values. In this case F will be certainly injective. In the cases we will consider, the number of nodes used in the local nodes sets will be slightly over 6 and we didn't encounter any non-invertible $\hat{A}$.

4. Practical Considerations

In this section we will treat some practical issues encountered in calculating and solving the discrete minimization problem.

4.1. **Calculation and storage of Voronoi diagrams.** For calculation of the Voronoi diagram we use the sweep algorithm of Fortune [6], which has a complexity of $\mathcal{O}(N\log N)$. Some code is available for this, but because we need to store the result in some other way than just a list of edges, a new code has been written, which uses the sweep-line algorithm but stores the diagram in a suitable data structure which is called the 'doubly-connected edge list'.

In such a data structure it is easy to find all neighboring edges, vertices and faces of a face. Because in our case a face is just a Voronoi tile and every tile represents exactly one node, we can find neighboring nodes in this way and also a list of vertices, making up the polygon which describes the Voronoi tile of a node. The algorithms involved in finding these neighboring elements are of complexity $\mathcal{O}\left(N^0\right)$. More on doubly-connected edge lists can be found in the book [4].

4.2. **Integrals of multinomials on polygons.** When for a given node i the neighboring nodes are found we have to calculate the matrices $(B_i B_i^T)^{-1} B_i$ in equation (17). To find the matrices M_i^A and M_i^L also the calculation of I_{Ω_i} is required.

To see how this can be done we assume that we want to integrate a function $f\colon \mathbb{R}^2 \to \mathbb{R}$, with $f(x,y) = x^n y^m$, on a polygon given by the points $\mathbf{r}_j$, $j = 1, \dots, N$, describing in counter clockwise order the polygon Ω_i. (Here the N is different from the one used earlier, which denoted the total number of nodes.) If we define

$$F(x,y) = \begin{pmatrix} \frac{x^{n+1} y^m}{n+1} \\ 0 \end{pmatrix}, \quad \nabla \cdot F(x,y) = f(x,y),$$

then, by using Gauss' divergence theorem,

$$\int_{\Omega_i} f \, d\mathbf{x} = \int_{\Omega_i} \nabla \cdot F \, d\mathbf{x} = \int_{\partial\Omega_i} F(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) \, d\mathbf{x} = \sum_{j=1}^{N} \int_{\partial\Omega_i^j} F(\mathbf{x}) \cdot \mathbf{n}_j \, d\mathbf{x},$$

where $\partial\Omega_i^j$ is the line segment $\mathbf{r}_j\mathbf{r}_{j+1}$ for $j \leq N$, $\partial\Omega_i^N = \mathbf{r}_N\mathbf{r}_1$, and $\mathbf{n}_j$ is the normal vector pointing outward. When using the parameterizations $\gamma_j\colon [0,1] \to \mathbb{R}^2$, with $\gamma_j(t) = \mathbf{r}_j + t\Delta\mathbf{r}_j$, where $\Delta\mathbf{r}_j = \mathbf{r}_{j+1} - \mathbf{r}_j$, we have $\|\gamma_j'(t)\| = \|\Delta\mathbf{r}_j\|$ and $\mathbf{n}_j = \frac{1}{\|\Delta\mathbf{r}_j\|}[\Delta y_j, -\Delta x_j]^T$, implying

$$\int_{\partial\Omega_i^j} F(\mathbf{x}) \cdot \mathbf{n}_j \, d\mathbf{x} = \int_0^1 \begin{pmatrix} \frac{x_j(t)^{n+1} y_j(t)^m}{n+1} \\ 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta y_j \\ -\Delta x_j \end{pmatrix} dt.$$

Consequently,

$$\int_{\Omega} x^n y^m \, d\mathbf{x} = \sum_{j=1}^{N} \frac{\Delta y_j}{n+1} \int_0^1 (x_j + t\Delta x_j)^{(n+1)} (y_j + t\Delta y_j)^m \, dt.$$

All entries of I_{Ω_j} have this form and they differ only in the choice of n and m.

4.3. **Solving the linear system.** Once we have built the matrices $\hat{A}$ and $\hat{L}$, using a sparse matrix structure, we have to solve the system

$$\hat{A}\mathbf{u} = \hat{L}\mathbf{f}.$$

Although the matrices $\hat{A}$ and $\hat{L}$ are sparse, they do not have a band structure in general. Due to the use of arbitrary nodes, the non-zero entries are actually scattered throughout the whole matrix. For solving the system by using an LU-decomposition, it would be advantageous to have a band structure, because then the number of non-zeros in the L and U matrices is also limited [9].

Fortunately, the sweep-line algorithm for calculation of the Voronoi diagram makes use of an ordering of the nodes that can be used here, because it renders our matrices into band structured matrices. The ordering is a lexicographical ordering, where the nodes are ordered on the basis of their coordinates such that

$$(x_1, y_1) \leq (x_2, y_2) \iff y_1 \leq y_2 \quad \text{or} \quad x_1 \leq x_2 \text{ and } y_1 = y_2.$$

Because nodes used in a local approximation are close to each other, the matrices will have the desired band structure after applying this ordering. In Figure 4 we see the sparsity structure of the matrix $\hat{A}$ for the node set of the left picture, using the original ordering (middle picture) and the ordering used by the sweep-line algorithm (right picture). In the original ordering the interior points are chosen randomly, while the boundary nodes are ordered along the boundary. This gives the pattern in the matrix as can be seen in the middle picture.

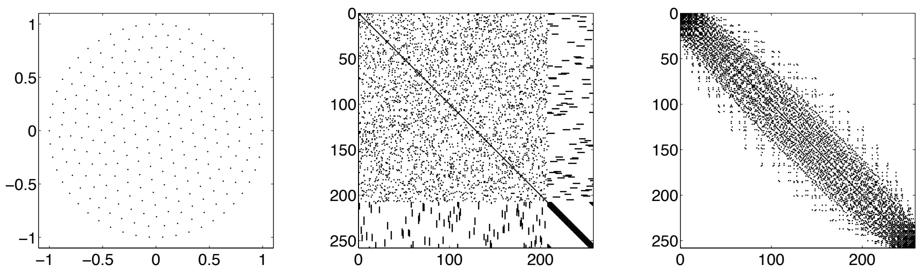

FIGURE 4. Left. Node set $\mathcal{N}$ consisting of 257 nodes. Middle. Sparsity structure of $\hat{A}$ and $\hat{L}$ with original ordering of $\mathcal{N}$. Right. Sparsity structure of $\hat{A}$ and $\hat{L}$ with ordering of $\mathcal{N}$ used by the Voronoi diagram calculation. The number of non-zero entries equals 5409.

5. Choosing nodes in the domain

In this section we will discuss an algorithm for choosing nodes in the domain. As said in the introduction we will work with very diverse domain geometries. Starting with a domain of which the boundary is piecewise smooth and which could have some holes, we need to put nodes in its interior and on the boundary.

Putting nodes on the boundary is relatively easy if we assume that the boundary is given by some parametrization. Given a number of nodes, we can distribute them over the boundary, possibly taking into account the curvature, so that to regions with high curvature comparatively many nodes are being assigned.

To put nodes in the interior is harder because testing whether a node is inside or outside a certain domain can be very difficult. Aside from this problem, there is also the problem of making a distribution of nodes that suits well function approximation and solving PDEs. As it turns out, the set of nodes one gets by randomly distributing points into a domain will show a lot of clustering, which is not optimal for function approximation and which will give also ill-conditioning problems when used for solving PDEs. Another issue is that we want to be able to impose some variation in the local node density, so that sufficiently many nodes are used in the neighborhoods of sources and on the boundaries and fewer at some distance of them.

5.1. **Lloyd's method.**

We will now discuss an algorithm to put nodes into the interior, assuming that there are nodes placed on the boundary already. The boundary nodes are connected by straight lines, transforming our domain into one which has a polygonal boundary. The first step is to find a rectangular region which lies entirely in the interior of the domain. The user of the algorithm has to find it by inspection of the domain.

In this rectangular region nodes are assigned in an arbitrary way. The rest of the algorithm consists of an alternating sequence of the following two steps.

Step 1. *Calculation of the Voronoi diagram.* Given the set of nodes, calculate the Voronoi diagram. The tiles Ω_i at the boundaries are cut off by the straight lines connecting the boundary nodes. This results in a tessellation of the polygonal domain, in which every tile has one node in it. An example has been shown in Figure 3. □

Step 2. *Node replacement using mass centroids.* Given the tessellation of the domain, shift every node to the mass centroid of the tile it is in, except for the nodes on the boundary. The mass centroid of a tile Ω_i is defined as

$$\mathbf{z}_i^{\text{centroid}} = \frac{\int_{\Omega_i} \mathbf{x}\, d\mathbf{x}}{\int_{\Omega_i} d\mathbf{x}}.$$

□

While alternating these steps the boundary nodes stay on the boundary and the interior nodes stay in the interior. The boundary nodes are fixed and thus also the polygonal boundary.

This algorithm is called Lloyd's method and more about it can be found in [5]. The basic idea behind it is that it tries to make all tiles equally large and spreads out all nodes into the domain while avoiding clustering. One might wonder whether the iteration converges or if the possibility exists that it will will run into some cycle. To make this somewhat clearer, we have the following theorem.

Theorem 1. *Suppose we are given an arbitrary disjoint node set $\{\mathbf{x}_i,\ i = 1, \ldots, N\}$ which is bounded by a piecewise linear boundary, consisting of a part of $\{\mathbf{x}_i\}$ connected by straight lines. Then any step in the alternating sequence of calculation of Voronoi diagrams and node replacement using mass centroids minimizes the functional*

$$F(\{\mathbf{x}_i\}, \{\Omega_i\}) = \sum_{i=1}^{N} \int_{\Omega_i} \|\mathbf{x}_i - \mathbf{x}\|^2\, d\mathbf{x}, \tag{22}$$

where $\{\Omega_i, i = 1, \ldots, N\}$ is a tiling, such that $\mathbf{x}_i \in \Omega_i$ for all $i = 1, \ldots, N$. Step 1 chooses Ω_i to minimize (22) *for fixed $\mathbf{x}_i$ and step 2 chooses $\mathbf{x}_i$ to minimize* (22) *for fixed Ω_i.*

Proof. Denote the set bounded by the piecewise linear boundary by $\mathcal{D}$, resulting in $\mathcal{D} = \cup_i \overline{\Omega}_i$.

Step 1: to prove the assertion for the Voronoi diagram calculation step, we consider an arbitrary $\mathbf{x} \in \mathcal{D}$. Clearly, the contribution of the area around this point to the functional is determined by the value $\|\mathbf{x} - \mathbf{x}_i\|^2$, where $\mathbf{x}_i$ is some node of $\mathcal{N}$. The fact that $\|\mathbf{x} - \mathbf{x}_i\|$ is minimal over all $i = 1, \ldots, N$ by definition of the Voronoi diagram, makes that $\|\mathbf{x} - \mathbf{x}_i\|^2$ is also minimal. Because this can be done for arbitrary $\mathbf{x} \in \mathcal{D}$, the functional is minimal over all possible tessellations of $\mathcal{D}$.

Step 2: the assertion can be proved by considering $\int_{\Omega_i} \|\mathbf{z} - \mathbf{x}\|^2\, d\mathbf{x}$ for a tile Ω_i and some arbitrary $\mathbf{z} \in \mathbb{R}^2$. To minimize the integral we can set the gradient with respect to $\mathbf{z}$ equal to zero. This results in

$$\int_{\Omega_i} (\mathbf{z} - \mathbf{x})\, d\mathbf{x} = 0 \implies \frac{\int_{\Omega_i} \mathbf{x}\, d\mathbf{x}}{\int_{\Omega_i} d\mathbf{x}}, \tag{23}$$

which is exactly the definition of the mass centroid. □

According to the theorem both steps minimize $F(\{\mathbf{x}_i\}, \{\Omega_i\})$ and therefore during the alternating procedure $F(\{\mathbf{x}_i\}, \{\Omega_i\})$ will be non-increasing. Also, the functional is bounded from below, making the sequence of $F(\{\mathbf{x}_i\}, \{\Omega_i\})$ convergent. As a result no cycling can occur but on the other hand convergence of the node set $\{\mathbf{x}_i\}$ itself is not guaranteed and the found minimal value of F does not need to be a global minimizer. In Figure 5 some iterations are shown of a node choosing process which resulted after 200 iterations in the node set shown in the first picture of Figure 4.

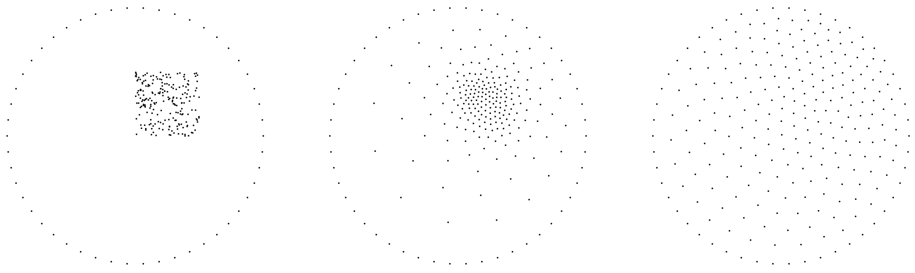
FIGURE 5. A node set at the start, after 10 iterations, and after 100 iterations.

5.2. **Adjusting local node density.** When using the method of the previous subsection we can get node distributions where neighboring nodes are on a more or less constant distance from each other. To impose some variation in local node density we can use a more general version of the algorithm which makes use of a density function in the evaluation of the centroids [5]. We take a different approach where after replacement of the nodes by the calculated centroids in step 2, we shift them a little. This shift is in the direction where a higher concentration of nodes is needed.

This procedure takes into account that variations in node density should be smooth rather than abrupt. The size of the shift is taken proportional to the size of the tile with respect to the direction of the shift, yielding that the node stays inside the tile. The direction of the shift is determined by attracting neighboring nodes. Given node i and attracting neighboring nodes j, the direction will be close to $\sum_j (\mathbf{x}_j - \mathbf{x}_i)$. The calculation of the shift follows the step of the replacement of the nodes to their mass centroids. To make it more precise we will now give a detailed description of the shifting step.

Step 3. *Node replacement by applying the shift.* Given the tile Ω_i, a set of vertices $\{\mathbf{y}_j\}$ and a set of attracting neighbors $\{\mathbf{x}_{i_k}\}$, first calculate the attracting direction $\mathbf{v} = \sum_k (\mathbf{x}_{i_k} - \mathbf{x}_i)$. Second, determine the minimal and maximal vertices with respect to this direction, i.e., $\mathbf{y}_{\min}$ minimizes and $\mathbf{y}_{\max}$ maximizes $\mathbf{y}_j \cdot \mathbf{v}$. Third, transform the tile using a transformation $(\xi, \eta) \to (x, y)$,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\mathbf{v}\cdot\mathbf{v}} \begin{pmatrix} v_1 & -v_2 \\ v_2 & v_1 \end{pmatrix} \begin{bmatrix} \mathbf{v}\cdot(\mathbf{y}_{\min} + \frac{1}{c}\log(\xi)(\mathbf{y}_{\max} - \mathbf{y}_{\min})) \\ \eta \end{bmatrix}, \tag{24}$$

where $c > 0$ is some constant determining the shift size with respect to the tile size. Finally, for the transformed tile the mass centroid is calculated and the result is transformed back, using the inverse transformation, to give the new location of the node. $\square$

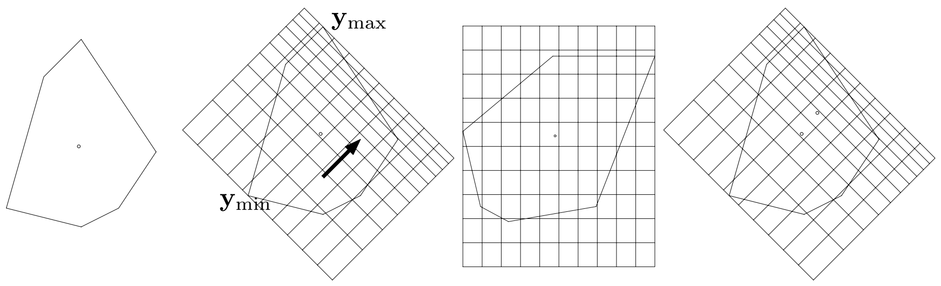

FIGURE 6. Shift calculation by transformation of the tile

Figure 6 illustrates the process of calculating the shift for an example tile, including the transformation involved. Here $c = 2$ and the direction of $\mathbf{v}$ is given by the arrow in the second picture. The first picture shows the tile and its mass centroid. The second picture adds the coordinate frame of the transformation, while the third picture displays the tile in the transformed coordinate system and its mass centroid with respect to this system. The last picture shows the tile with the original mass centroid and the shifted point.

Attracting neighboring nodes. To determine how the nodes attract each other all nodes are classified by some integer value. Attraction can than be implemented by defining the attracting neighboring nodes of a node as the neighbors that have a integer value which is higher than their own integer value.

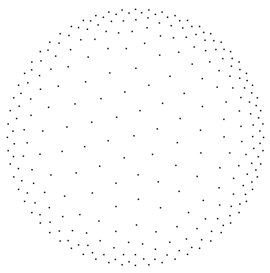

FIGURE 7. Example of refinement near the boundary

Lets us for example assume that we would like to have refinement near the boundary. Then all boundary nodes could be classified by 2, all neighbors of boundary nodes by 1, and the rest by 0. By cycling through the steps: 1. calculation of the Voronoi diagram, 2. giving every node a type, 3. calculation of mass centroids, 4. calculation of shifts, the global nodes would gradually change in a node set which has some refinement near the boundary. The parameter c specifies the maximal spatially decay in the distances between the nodes in a refinement area. The number of different types specifies the size of a refinement area.

In Figure 7 a refinement near the boundary is achieved by classifying boundary nodes as 3, their neighbors as 2 and their neighbors' neighbors as 1.

6. Numerical Tests

In this section we will carry out two numerical tests. We will start with a convergence test on the unit circle where we use a uniform distribution of nodes. I.e., after inserting the nodes randomly in the domain we use the iteration procedure from Section 5 with a constant node density. We calculate the solution of the elliptic problem (14) for the source function

$$f(r,\theta) = \frac{2\pi}{r(\pi^2-4)}\cos(\tfrac{1}{2}\pi r)\Big((\pi^2+1)r\cos(\tfrac{1}{2}\pi r) + \pi\sin(\tfrac{1}{2}\pi r)\Big),$$

with r and θ polar coordinates. With $D = 1$ and $\kappa = 1$ the exact solution is

$$u(r,\theta) = \frac{\pi}{\pi^2-4}\Big(2\cos^2(\tfrac{1}{2}\pi r) + \pi^2\Big).$$

Figure 8 shows both the source function (left) and the solution (right).

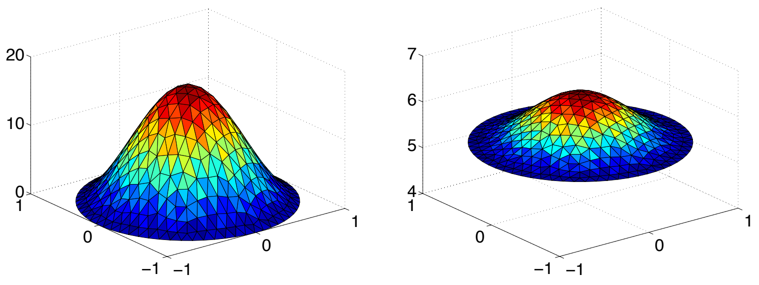

Figure 8. source function and solution of equation (14)

For the test the solution is calculated thirty times. The number of nodes is increased every time, such that the maximal distance between two neighboring nodes $\Delta_{\max}$ will vary gradually between 0.2 and 0.02. The maximal local radius h, used in the convergence analysis, will be around twice this distance and will therefore also change with a factor 10. The number of nodes used varies between 106 and 9226.

For each numerical solution we computed the error $\mathbf{e} = \mathbf{u}_{\text{num}} - \mathbf{u}_{\text{exact}}$, its L^2-norm $\|\mathbf{e}\|_{L^2} = (\mathbf{e}^T\hat{L}\,\mathbf{e})^{1/2}$, and its maximum error $\|\mathbf{e}\|_\infty$. In Figure 9 shows the results of the test and both errors display a second order convergence.

In the second test we calculate the solution of equation (14), where the domain is the unit circle with a hole in it. The source function is formed by two narrow

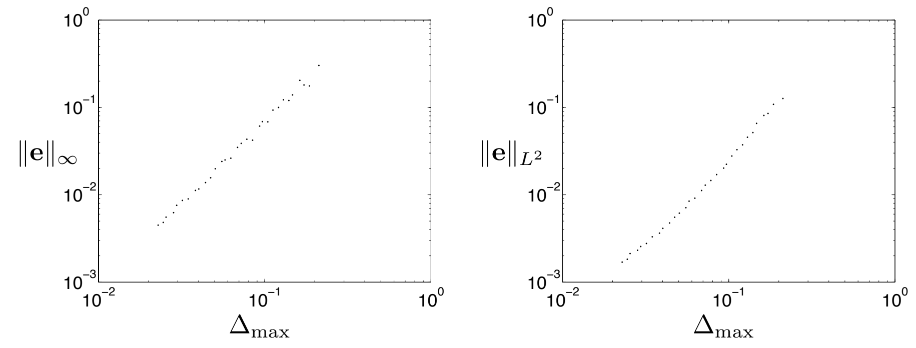

FIGURE 9. Maximum norm and L^2-norm errors against $\Delta_{\max}$.

peaks somewhere in the domain. The peaks are circle symmetric with a circular support of radius of $\ell = 0.02$, inside of which they are given by

$$f(r,\phi) = \frac{2\pi}{\ell^2(\pi^2-4)}\cos^2(\tfrac{1}{2\ell}\pi r),$$

with (r,ϕ) being polar coordinates centered at the location of the peak. Again we take $D = 1$ and $\kappa = 1$.

The refinement strategy is used to put a high concentration of nodes in the neighborhood of the peaks and near the boundaries. With 1890 nodes in total this yields the left picture in Figure 10. The right picture shows a magnification around the support of one of the peaks. In such a circular support 70 nodes are being used. To determine the nodes, first the nodes for the peaks and the boundary nodes are determined, after which they are fixed. Then the other nodes are added and the node shifting iterations are done. Here the nodes for the peaks are surrounded with eight rings of attracting nodes, while for the boundaries three and five rings are used, respectively. In Figure 11 the numerical solution is shown. For the integral

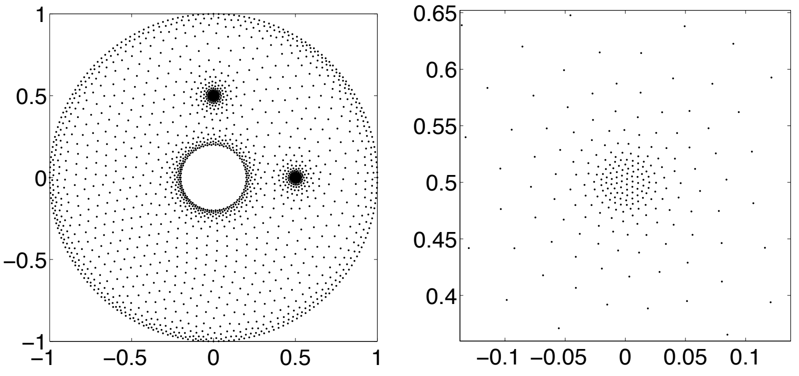

FIGURE 10. node distribution (left) and refinement around one of the peaks (right).

of the solution we have

$$\int_\Omega u(\mathbf{x})\,d\mathbf{x} = \int_\Omega f(\mathbf{x})\,d\mathbf{x} = 2.$$

A second order approximation of this integral is $\mathbf{1}^T\hat{L}\mathbf{u} = 1.9875$, where $\mathbf{1}$ is a vector whose entries are all equal to 1. To fill the domain with nodes such that the node density would be equal to the node density as it is in the peak support, would require over 100.000 nodes. We did a similar experiment with peak widths ten times as small as in the test under consideration, but with the same number of nodes. With the number of rings of attracting nodes changed from 8 to 13, the result was $\mathbf{1}^T\hat{L}\mathbf{u} = 1.96$. In that case a node distribution with a uniform node density would require over 10 million nodes.

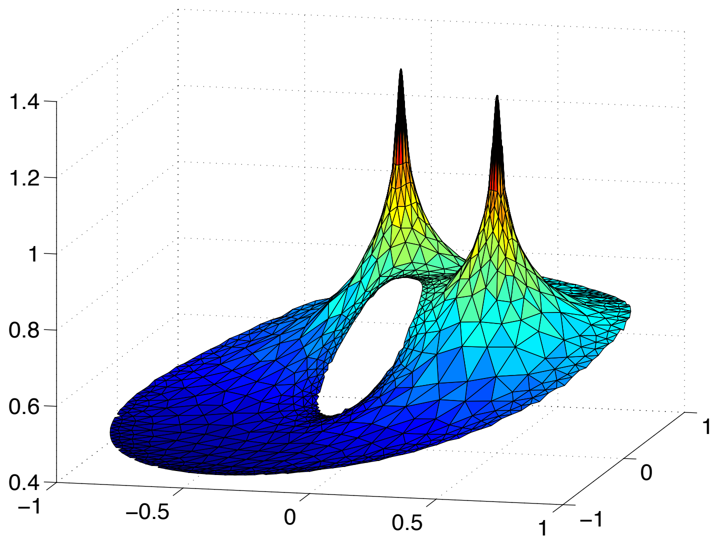

FIGURE 11. Numerically computed peaks

7. SUMMARY

In this paper we developed a meshfree method for solving time-discrete diffusion equations that arise in equation systems used in models from brain research. Important criteria for a suitable method are flexibility with respect to domain geometry and easy refinement possibilities. Both criteria are met when using a meshfree method. The two main results of this paper are a meshfree discretization of the modified Helmholtz operator and a node choosing algorithm that allows for easy placement of nodes into a given domain while varying node density. Both the discretization and the node choosing algorithm use a Voronoi diagram based on the given node set.

The meshfree discretization uses a Voronoi diagram for finding neighboring nodes of a node and for approximation of an integral on the domain. It is based on a local least squares approximation and the minimization problem in H^1 that is related to the modified Helmholtz equation in combination with the boundary conditions. The minimization problem is discretized by using node functions instead of elements of H^1. The node choosing algorithm uses a Voronoi diagram for shifting nodes in the right direction. Here the final node distribution tends to be optimal in a certain sense. During the algorithm the nodes repel each other, thereby resulting in some kind of uniformity.

The local least squares approximation underlying the discretization uses a finite number of nodes, called the local node set, to determine a local approximation of a function. Its convergence in the maximum norm is of second order in the diameter of the local node set, if the quality of this set is sufficient, the latter being measured by a determinant based on the set. When using the node choosing algorithm, the numerical solution of the diffusion equation converges in second order in the maximal diameter of all used local node sets.

In a convergence test the second order convergence is displayed and the method is applied to the modified Helmholtz equation on a circular domain with a hole in it and a source function with very small support compared to the domain.

References

[1] T. Belytschko, Y. Krongauz, D. Organ, M.Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering, special issue on Meshless Methods*, 139:3–47, 1996.

[2] T. Belytschko, Y.Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[3] Y.J. Choi and S.J. Kim. Node generation scheme for the meshfree method by voronoi diagram and weighted bubble packing. Fifth U.S. National Congress on Computational Mechanics, Boulder, CO, 1999.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry.* Springer-Verlag, 2nd edition, 2000.

[5] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[6] S. Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[7] T.P. Fries and H.G. Matthies. Classification and overview of meshfree methods. Informatikbericht 2003-03, Institute of Scientific Computing, Technical University Braunschweig, Brunswick, Gernmany,, 2003.

[8] D. Gilbarg and N.S. Trudinger. *Elliptic Partial Differential Equations of Second Order.* Springer Verlag, 2001.

[9] G.H. Golub and C.F. van Loan. *Matrix Computations.* The John Hopkins University Press, 3rd edition, 1996.

[10] H.G.E. Hentschel and A. van Ooyen. Models of axon guidance and bundling during development. *Proc. R. Soc. Lond. B.*, 266:2231–2238, 1999.

[11] R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[12] S.R. Idelsohn, E. O nate, F. Del Pin, and N. Calvo. The meshless finite element method. *Int. J. Numer. Methods Eng.*, 2001.

[13] J.K. Krottje. On the dynamics of a mixed parabolic-gradient system. *Communications on Pure and Applied Analysis*, 2003. (To appear).

[14] Shaofan Li and Wing Kam Liu. Meshfree and particle methods and their applications. *Applied Mechanics Review*, 55:1–34, 2002.

[15] X.-Y. Li, S.-H. Teng, and A. Üngör. Point placement for meshless methods using sphere packing and advancing front methods. Technical report, University of Illinois at Urbana-Champaign, 2000.

[16] B. Nayroles, G. Touzet, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Comp. Mech.*, 10:307–318, 1992.

[17] N. Sukumar, B. Moran, A. Yu Semenov, and V.V. Belikov. Natural neighbour galerkin methods. *International Journal for Numerical Methods in Engineering*, (50):1–27, 2001.

[18] E.W. Weisstein. *Concise encyclopedia of mathematics.* CRC Press, 2nd edition, 2002.