A 3D Efficient Procedure for Shepard Interpolants on Tetrahedra

A 3D efficient procedure for Shepard interpolants on tetrahedra

Roberto Cavoretto[0000−0001−6076−4115],
Alessandra De Rossi*[0000−0003−1285−3820],
Francesco Dell'Accio[0000−0003−4879−894X],
Filomena Di Tommaso[0000−0002−4638−2994]

Department of Mathematics "Giuseppe Peano", University of Torino
Via Carlo Alberto 10, 10123 Torino, Italy
Department of Mathematics and Computer Science, University of Calabria
via P. Bucci, Cubo 30A, 87036 Rende (CS), Italy
roberto.cavoretto@unito.it, alessandra.derossi@unito.it,
francesco.dellaccio@unical.it, ditommaso@mat.unical.it

Abstract. The need of scattered data interpolation methods in the multivariate framework and, in particular, in the trivariate case, motivates the generalization of the fast algorithm for triangular Shepard method. A block-based partitioning structure procedure was already applied to make the method very fast in the bivariate setting. Here the searching algorithm is extended, it allows to partition the domain and nodes in cubic blocks and to find the nearest neighbor points that need to be used in the tetrahedral Shepard interpolation.

Keywords: scattered data interpolation· tetrahedral Shepard operator· fast algorithms· approximation algorithms.

1 Introduction

Given a set of values of a function f at certain scattered nodes $X_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ in a compact convex domain $\Omega \subset \mathbb{R}^2$, the triangular Shepard method [8] can be applied efficiently to interpolate the target function $f : \Omega \to \mathbb{R}$. In [5] we proposed a triangular Shepard method which combines triangle-based basis functions with linear combinations of the values $f(\boldsymbol{x}_i)$ at the vertices of the triangles. Moreover, the triangulation can be found in an efficient way by reducing the number of triangles. The triangulation considered is called *compact triangulation* and it allows the triangles to overlap or being disjoint. These triangulations are determined by minimizing the bound of the error of the linear interpolant on the vertices of the triangle, chosen in a set of nearby nodes. For these triangulations a block-based partitioning structure procedure was presented in [3] to make the method very fast, since the vertices of the triangles must be chosen in a set of nearby nodes.

In recent years an increasing attention to the multivariate framework was given. For this reason we propose in this paper a generalization to the 3D setting.

More precisely, we propose a fast searching procedure to apply to the tetrahedral Shepard interpolation. It allows to partitioning the 3D domain and nodes in cubic blocks and to find the nearest neihbor points to compute the Shepard interpolant on tetrahedra. Similar algorithms were also analized in [2] in the context of trivariate partition of unity methods combined with the use of local radial kernels.

The paper is organized as follows. In Section 2 the tetrahedral Shepard method for trivariate interpolation is recalled. In Section 3 we give a pseudo-code of the complete interpolation algorithm, presenting the procedures used to identify and search the nearest neighbor points in the 3D interpolation scheme. In Section 4 we show some numerical experiments obtained to illustrate the performance of our tetrahedral Shepard algorithm. Finally, Section 5 contains conclusions and future work.

2 Tetrahedral Shepard Interpolant

Let be $X_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ a set of data points or nodes of $\mathbb{R}^3$ with an associated set of function data $F_n = \{f_1, \ldots, f_n\}$ and $H = \{h_1, \ldots, h_m\}$ a set of tetrahedra with vertices in X_n. Let us denote by $W_j = \{\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \boldsymbol{x}_{j_3}, \boldsymbol{x}_{j_4}\}$ the set of vertices of h_j, $j = 1, \ldots, m$. Moreover, we assume that the set $\{W_j\}_{j=1,\ldots,m}$ constitutes a cover of X_n, that is

$$\bigcup_{j=1}^{m} W_j = X_n.$$

We can associate to each tetrahedra h_j the set of barycentric coordinates of a point $\boldsymbol{x} \in \mathbb{R}^3$, that is

$$\mu_{j,j_1}(\boldsymbol{x}) = \frac{W(\boldsymbol{x}, \boldsymbol{x}_{j_2}, \boldsymbol{x}_{j_3}, \boldsymbol{x}_{j_4})}{W(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \boldsymbol{x}_{j_3}, \boldsymbol{x}_{j_4})}, \quad \mu_{j,j_2}(\boldsymbol{x}) = \frac{W(\boldsymbol{x}_{j_1}, \boldsymbol{x}, \boldsymbol{x}_{j_3}, \boldsymbol{x}_{j_4})}{W(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \boldsymbol{x}_{j_3}, \boldsymbol{x}_{j_4})},$$

$$\mu_{j,j_3}(\boldsymbol{x}) = \frac{W(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \boldsymbol{x}, \boldsymbol{x}_{j_4})}{W(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \boldsymbol{x}_{j_3}, \boldsymbol{x}_{j_4})}, \quad \mu_{j,j_4}(\boldsymbol{x}) = \frac{W(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \boldsymbol{x}_{j_3}, \boldsymbol{x})}{W(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2}, \boldsymbol{x}_{j_3}, \boldsymbol{x}_{j_4})},$$

where $W(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}, \boldsymbol{z})$ denotes $\frac{1}{6}$ the signed volume of the tetrahedra h_j. The linear polynomial $\lambda_j(\boldsymbol{x})$ which interpolates the data at the vertices of the tetrahedra h_j can be expressed in terms of barycentric coordinates in the following form

$$\lambda_j(\boldsymbol{x}) = \sum_{k=1}^{4} \mu_{j,j_k}(\boldsymbol{x}) f_{j_k}, \quad j = 1, \ldots, m. \tag{1}$$

The tetrahedral basis functions are a normalization of the product of the inverse distances from the vertices of the tetrahedra h_j

$$\beta_{\nu,j}(\boldsymbol{x}) = \frac{\prod_{\ell=1}^{4} ||\boldsymbol{x} - \boldsymbol{x}_{j_\ell}||^{-\nu}}{\sum_{k=1}^{m} \prod_{\ell=1}^{4} ||\boldsymbol{x} - \boldsymbol{x}_{k_\ell}||^{-\nu}}, \qquad j = 1, \ldots, m, \quad \nu > 0, \tag{2}$$

where $||\cdot||$ is the Euclidean norm. The tetrahedral Shepard method is defined by

$$\mathcal{T}_\nu[f](\boldsymbol{x}) = \sum_{j=1}^{m} \beta_{\nu,j}(\boldsymbol{x})\,\lambda_j(\boldsymbol{x}). \tag{3}$$

Tetrahedral basis functions form a partition of unity, as the triangular Shepard ones, and allow the interpolation of functional and derivative values. In fact, the following results hold, the proofs can be easily obtained in analogy with [5, Proposition 2.1].

Proposition 1. *The tetrahedral basis function $\beta_{\nu,j}(\boldsymbol{x})$ and its gradient (that exists for $\nu > 1$) vanish at all nodes $\boldsymbol{x}_i \in X_n$ that are not a vertex of the corresponding tetrahedron h_j. That is,*

$$\beta_{\nu,j}(\boldsymbol{x}_i) = 0, \tag{4}$$

$$\nabla\beta_{\nu,j}(\boldsymbol{x}_i) = 0, \quad \nu > 1, \tag{5}$$

for any $j = 1,\dots,m$ and $i \notin \{j_1, j_2, j_3, j_4\}$. Moreover, they form a partition of unity, that is

$$\sum_{j=1}^{m} \beta_{\nu,j}(\boldsymbol{x}) = 1 \tag{6}$$

and consequently, for each $i = 1,\dots,n$,

$$\sum_{j\in J_i} \beta_{\nu,j}(\boldsymbol{x}_i) = 1, \tag{7}$$

$$\sum_{j\in J_i} \nabla\beta_{\nu,j}(\boldsymbol{x}_i) = 0, \quad \nu > 1, \tag{8}$$

where $J_i = \big\{k \in \{1,\dots,m\} : i \in \{k_1, k_2, k_3, k_4\}\big\}$ is the set of tetrahedra which have $\boldsymbol{x}_i$ as a vertex.

These properties imply that the operator $\mathcal{T}_\nu$ satisfies the following ones, see [4] for details.

Proposition 2. *The operator $\mathcal{T}_\nu$ is an interpolation operator, that is,*

$$\mathcal{T}_\nu[f](\boldsymbol{x}_i) = f_i, \qquad i = 1,\dots,n,$$

and reproduces polynomials up to the degree 1.

The procedure to select the compact $3D$-triangulation (by tetrahedra) of the node set X_n strongly affects the results of the analysis of the convergence of the operator $\mathcal{T}_\nu[f](\boldsymbol{x})$.

In order to determine the approximation order of the tetrahedral operator, we denote by $\Omega \subset \mathbb{R}^3$ a compact convex domain containing X_n and by $C^{1,1}(\Omega)$ the class of differentiable functions $f : \Omega \to \mathbb{R}$ whose partial derivative of order 1 are Lipschitz-continuous, equipped with the seminorm

$$\|f\|_{1,1} = \sup\left\{\frac{\|D^\mu f(\boldsymbol{u}) - D^\mu f(\boldsymbol{v})\|}{\|\boldsymbol{u}-\boldsymbol{v}\|} : \boldsymbol{u},\boldsymbol{v} \in \Omega, \boldsymbol{u} \neq \boldsymbol{v}, \|\mu\| = 1\right\}. \tag{9}$$

We also denote by $\boldsymbol{e}_{k,\ell} = \boldsymbol{x}_{j_k} - \boldsymbol{x}_{j_\ell}$, with $k, \ell = 1, 2, 3, 4$, the edge vectors of the tetrahedron h_j. Then, the following result holds (for the proof see [4]).

Proposition 3. *Let $f \in C^{1,1}(\Omega)$ and $h_j \in H$ a tetrahedron of vertices $\boldsymbol{x}_{j_1}$, $\boldsymbol{x}_{j_2}$, $\boldsymbol{x}_{j_3}$, $\boldsymbol{x}_{j_4}$. Then, for all $\boldsymbol{x} \in \Omega$ we have*

$$|f(\boldsymbol{x}) - \lambda_j(\boldsymbol{x})| \leq ||f||_{1,1} \left(3\,||\boldsymbol{x} - \boldsymbol{x}_{j_1}||_2^2 + \frac{27}{2} C_j k_j\,||\boldsymbol{x} - \boldsymbol{x}_{j_1}||_2 \right), \tag{10}$$

where $k_j = \max_{k,\ell=1,2,3,4} \|\boldsymbol{e}_{k,\ell}\|$ and C_j is given by the ratio between the maximum edge and the volume, and then is a constant which depends only on the shape of the tetrahedron h_j. The error bound is valid for any vertex.

3 Trivariate Shepard Interpolation Algorithm on Tetrahedra

In this section we present the interpolation algorithm, which performs the tetrahedral Shepard method (3) using the block-based partitioning structure and the associated searching procedure. Here we consider $\Omega = [0, 1]^3$.

`INPUTS:` n, number of data; $X_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, set of data points; $F_n = \{f_1, \ldots, f_n\}$, set of data values; n_e, number of evaluation points; n_w, localizing parameter.

`OUTPUTS:` $E_{n_e} = \{\mathcal{T}_\nu[f](\boldsymbol{z}_1), \ldots, \mathcal{T}_\nu[f](\boldsymbol{z}_{n_e})\}$, set of approximated values.

`Step 1:` Generate a set $Z_{n_e} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_e}\} \subseteq \Omega$ of evaluation points.

`Step 2:` For each point $\boldsymbol{x}_i$, $i = 1, \ldots, n$, construct a neighborhood of radius

$$\delta = \frac{\sqrt{3}}{d}, \quad \text{with} \quad d = \left\lfloor \left(\frac{n}{8}\right)^{1/3} \right\rfloor.$$

where the value of d is suitably chosen extending the definition contained in [2]. This phase performs the localization.

`Step 3:` Compute the number b of blocks (along one side of the unit cube Ω) defined by

$$b = \left\lceil \frac{1}{\delta} \right\rceil.$$

In this way we get the side of each cubic block is equal to the neighborhood radius. This choice enables us to examine in the searching procedure only a small number of blocks, so to reduce the computational cost as compared to the most advanced searching techniques, as for instance the kd-trees [10]. The benefit is proved by the fact that this searching process is carried out in constant time, i.e. $O(1)$. Further, in this partitioning phase we number the cube-shaped blocks from 1 to b^3.

`Step 4:` Build the partitioning structure on the domain Ω and split the set X_n of interpolation nodes in b^3 cubic blocks. Here we are able to obtain a fast searching procedure to detect the interpolation points nearest to each of nodes.

`Step 5:` For each neighborhood or point (i.e., the neighborhood centre), solve the containing query and the range search problems to detect all nodes X_{n_k}, $k = 1, \ldots, b^3$, belonging to the k-th block and its twenty-six neighboring blocks (or less in case the block lies on the boundary). This is performed by repeatedly using a *quicksort* routine.

`Step 6:` For each data point $\boldsymbol{x}_i \in X_n$, fix its n_w nearest neighbors $\mathcal{N}(\boldsymbol{x}_i) \subset X_n$. Among the

$$\frac{n_w\,(n_w-1)\,(n_w-2)}{6}$$

tetrahedra with a vertex in $\boldsymbol{x}_i$, name it $\boldsymbol{x}_{j_1}$ and other three vertices in $\mathcal{N}(\boldsymbol{x}_i)$, choose the one which locally reduces the bound for the error of the local linear interpolant

$$3\,||\boldsymbol{x}-\boldsymbol{x}_{j_1}||_2^2+\frac{27}{2}k_j\frac{k_j^3}{W\left(\boldsymbol{x}_{j_1},\boldsymbol{x}_{j_2},\boldsymbol{x}_{j_3},\boldsymbol{x}_{j_4}\right)}\,||\boldsymbol{x}-\boldsymbol{x}_{j_1}||_2\,.$$

`Step 7:` Compute the local basis function $\beta_{\nu,j}(\boldsymbol{z})$, $j = 1, \ldots, m$, at each evaluation point $\boldsymbol{z} \in Z_{n_e}$.

`Step 8:` Compute the linear interpolants $\lambda_j(\boldsymbol{z})$, $j = 1, \ldots, m$, at each evaluation point $\boldsymbol{z} \in Z_{n_e}$.

`Step 9:` Apply the tetrahedral Shepard method (3) and evaluate the trivariate interpolant at the evaluation points $\boldsymbol{z} \in Z_{n_e}$.

4 Numerical Results

We present here accuracy and efficiency results of the trivariate interpolation algorithm proposed. The algorithm was implemented in MATLAB. All the numerical experiments have been carried out on a laptop with an Intel(R) Core i7 6500U CPU 2.50GHz processor and 8.00GB RAM.

In the following we analize the results obtained about several tests carried out. We solved very large interpolation problems by means of the tetrahedral Shepard method (3). To do this we considered two different distributions of irregularly distributed (or scattered) nodes contained in the unit cube $\Omega = [0,1]^3 \subset \mathbb{R}^3$, and taking a number n of interpolation nodes that varies from 2 500 to 20 000. More precisely, as interpolation nodes we focus on a few sets of uniformly random Halton points generated through the MATLAB program `haltonseq.m` [6], and pseudo-random points obtained by using the `rand` MATLAB command. In addition, the interpolation errors are computed on a grid consisting of $n_e = 21 \times 21 \times 21$ evaluation points, while as localizing parameter we fix the value $n_w = 13$ and $\nu = 2$.

In the various experiments we discuss the performance of our interpolation algorithm assuming the data values are given by the following two trivariate test functions:

$$f_1(x_1, x_2, x_3) = \cos(6x_3)(1.25 + \cos(5.4x_2))/(6 + 6(3x_1 - 1)^2),$$
$$f_2(x_1, x_2, x_3) = \exp(-81/16((x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2))/3.$$

These functions are usually used to test and validate new approximation methods and algorithms (see e.g. [9]).

As a measure of the accuracy of our results, we compute the Maximum Absolute Error (MAE) and the Root Mean Square Error (RMSE), whose formulas are respectively given by

$$\text{MAE} = ||f - \mathcal{T}_\nu[f]||_\infty = \max_{1 \le i \le n_e} |f(\boldsymbol{z}_i) - \mathcal{T}_\nu[f](\boldsymbol{z}_i)|$$

and

$$\text{RMSE} = \frac{1}{\sqrt{n_e}} ||f - \mathcal{T}_\nu[f]||_2 = \sqrt{\frac{1}{n_e} \sum_{i=1}^{n_e} |f(\boldsymbol{z}_i) - \mathcal{T}_\nu[f](\boldsymbol{z}_i)|^2},$$

where $\boldsymbol{z}_i \in Z_{n_e}$ is an evaluation point belonging to the domain Ω.

In Tables 1–2 we report MAEs and RMSEs that decrease when the number n of interpolation points increases. Comparing then the errors obtained by using the two data distributions, we can note that a (slightly) better accuracy is achieved whenever we employ Halton nodes. This fact is basically due to greater level of regularity of Halton points than pseudo-random MATLAB nodes. Analyzing the error behavior with the test functions f_1 and f_2, we get similar results in terms of accuracy of the interpolation scheme.

	f_1		f_2	
n	MAE	RMSE	MAE	RMSE
2 500	4.29E−2	4.63E−3	1.37E−2	1.99E−3
5 000	3.75E−2	3.04E−3	1.03E−2	1.14E−3
10 000	2.39E−2	2.05E−3	5.96E−3	7.33E−4
20 000	1.72E−2	1.33E−3	3.41E−3	4.50E−4

Table 1. MAE and RMSE computed on Halton points.

Then we compare the performance of the optimized searching procedure based on the partitioning of nodes in cubic blocks with a standard implementation of the algorithm where one computes all the distances between the interpolation nodes. With regard to the efficiency of the 3D Shepard interpolation

	f_1		f_2	
n	MAE	RMSE	MAE	RMSE
2 500	6.56E−2	5.49E−3	2.28E−2	2.49E−3
5 000	3.89E−2	3.89E−3	1.62E−2	1.63E−3
10 000	4.00E−2	2.71E−3	8.86E−3	1.03E−3
20 000	1.77E−2	1.65E−3	7.60E−3	6.38E−4

Table 2. MAE and RMSE computed on pseudo-random MATLAB points.

algorithm the CPU times computed in seconds are around 27 and 55 for the sets of 10000 and 20000 nodes, respectively. Using a standard implementation the seconds increase to about 41 and 318 for the two sets. From this study we highlight a remarkable enhancement in terms of computational efficiency when the new partitioning and searching techniques are applied.

5 Conclusions and Future Work

In this paper we presented a new trivariate algorithm to efficiently interpolate scattered data nodes using the tetrahedral Shepard method. Since this interpolation scheme needs to find suitable tetrahedra associated with the nodes, we proposed a fast searching procedure based on the partitioning of domain in cube blocks. Such a technique turned out to be computationally more efficient than a standard one. Numerical experiments showed good performance of our procedure, which enabled us to quickly deal with a large number of nodes.

Another possible extension is given by a spherical triangular or tetrahedral Shepard method, which can be applied on the sphere $\mathbb{S}^2$ or other manifolds (see e.g. [1,11]).

Acknowledgments

The authors acknowledge support from the Department of Mathematics "Giuseppe Peano" of the University of Torino via Project 2019 "Mathematics for applications". Moreover, this work was partially supported by INdAM – GNCS Project 2019 "Kernel-based approximation, multiresolution and subdivision methods and related applications". This research has been accomplished within RITA (Research ITalian network on Approximation).

References

1. Allasia, G., Cavoretto, R., De Rossi, A.: Hermite-Birkhoff interpolation on scattered data on the sphere and other manifolds. Appl. Math. Comput. **318**, 35–50 (2018)

2. Cavoretto R., De Rossi A., Perracchione E.: Efficient computation of partition of unity interpolants through a block-based searching technique. Comput. Math. Appl. **71**, 2568–2584 (2016)
3. Cavoretto R., De Rossi A., Dell'Accio F., Di Tommaso F.: Fast computation of triangular Shepard interpolants. J. Comput. Appl. Math. **354**, 457–470 (2019)
4. Cavoretto R., De Rossi A., Dell'Accio F., Di Tommaso F.: An efficient trivariate algorithm for tetrahedral Shepard interpolation, submitted (2019)
5. Dell'Accio F., Di Tommaso F., Hormann, K.: On the approximation order of triangular Shepard interpolation. IMA J. Numer. Anal. **36**, 359–379 (2016)
6. Fasshauer, G.E.: Meshfree Approximation Methods with MATLAB, World Scientific Publishing Co., Singapore (2007)
7. Fasshauer, G.E., McCourt, M.J.: Kernel-based Approximation Methods using MATLAB. Interdisciplinary Mathematical Sciences, vol. 19, World Scientific Publishing Co., Singapore (2015)
8. Little F. F.: Convex combination surfaces. In *Surfaces in Computer Aided Geometric Design* (ed. Barnhill R. E., Boehm W.), Amsterdam (North-Holland), 99–108 (1983)
9. Renka, R. J.: Multivariate Interpolation of Large Sets of Scattered Data. ACM Trans. Math. Softw. **14**, 139–148 (1988)
10. Wendland, H.: Scattered Data Approximation. Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge (2005)
11. Zhang, M. and Liang, X.-Z. On a Hermite interpolation on the sphere. Appl. Numer. Math. **61**, 666–674 (2011).