Multidimensional Scattered Time-varying Scattered Data Meshless Interpolation for Sensor Networks

Based on NATO Modelling&Simulation Centre of Excellence talk-MESAS 2022

Vaclav Skala[1]

[1]Dept.of Computer Science and Engineering
Faculty of Applied Sciences
University of West Bohemia, Pilsen
Czech Republic

ICCSA 2023

Pilsen City

(a) St.Bartolomew Church 1295

(b) FAV building

(c) The 2nd largest in Europe

Pilsen City

(a) Radbusa river & Museum

(b) even dogs drink a beer

Plzen is an old city [first records of Plzen castle 976] city of culture, industry, and brewery

City, where today's beer fermentation process was invented that is why today's beers are called Pilsner - world wide

"Real science" in the XXI century

The mysterious castle in the Carpathians [1]

[1] Courtesy of the Czech Film, Barrandov

Introduction

Data interpolation and approximation is a frequent task in many areas. Usually the interpolation is used for data sets $h_i = f(\mathbf{x_i})$, where h_i is a value at $\mathbf{x_i} \in \Omega$, where Ω is the data domain in E^d and $d = 2, 3$ mostly. The data domain is somehow tessellated, but not necessarily by the Delaunay triangulation(DT). The values h_i might be scalar or vector values, e.g. a wind velocity (v_x, v_y, v_z). In the case of spatio-temporal data, the "framing" is implicitly expected with the fixed known correspondence of points.

It leads to spatio-temporal meshes with the fixed connectivity of points in frames t_i and t_{i+1}, if the domain tessellation is used. It should be noted that if the spatial domain $\Omega \in E^3$ then in the spatio-temporal case a tessellation for E^4 is to be used.

Introduction

A usual tessellation technique is the Delaunay triangulation(DT). However, the computational complexity is $O(n^{\lceil d/2+1 \rceil})$, i.e. $O(n^2)$ for $d = 2$, $O(n^3)$ for $d = 3,4$. Due to numerical robustness issues, the complexity of the DT implementation grows significantly with the dimensionality. Also, the smoothness of the final interpolation of the h_i values is a fundamental requirement, which is not an easy task if triangular or tetrahedral meshes are used to represent the data domain. This approach can be used in the spatio-temporal case, when the data are "framed" for the given "time-slice" t_i and with fixed and known connectivity of points.

However, there are many areas when an interpolation of scattered spatio-temporal "non-framed" data are required, e.g. sensor networks, floating buoys with sensors on sea, when data sources are not constantly on-line and sending data occasionally only, e.g. tsunami detection, ships and submarine identification. Such sensors are connected only shortly. It leads to energy savings and hard detection of those.

Spatio-temporal data classification

Interpolation and approximation is usually made for the "ordered" data domains, e.g. rectangular, triangular and tetrahedral meshes, etc. In the CAD systems, the parametric space is used for interpolation, e.g. for parametric curves and surfaces.

Spatio-temporal data classification

The data domain can be classified as:

- ordered
  - structured
    - regular, e.g. a rectangular mesh where all elements have same size, triangular meshes with a constant vertex valency,
    - irregular, e.g. a rectangular mesh, but elements have different size triangular meshes with non-constant vertex valencies,
  - unstructured, e.g. general triangular or tetrahedral meshes,
- unordered
  - clustered - points form clusters in the data domain,
  - scattered - points are scattered across the domain, generally.

Spatio-temporal data classification

		Temporal property	
		Static	Dynamic
Spatial property	Static	$h = f(\mathbf{x})$	$h = f(\mathbf{x}, t)$
	Dynamic	$h = f(\mathbf{x}(t))$	$h = f(\mathbf{x}(t), t)$

Table: Classification of spatio-temporal data sets

The domain data can also be classified as static or dynamic in space and time, see Tab.1. It can be seen that the case $h = f(\mathbf{x}(t), t)$ represents interpolation of a scalar value h on the d-dimensional domain of $\mathbf{x}(t) \in \Omega(t)$, where the position $\mathbf{x}(t)$ is changing within time t. It should be noted that the $\Omega(t)$ is not constant in time, generally.

Spatio-temporal data classification

Now, the case of the scattered spatio-temporal domain is dynamic in both, i.e. $h(t) = f(\mathbf{x}(t), t)$, can be classified further as:

- framed in time - points lie on a hyperplane $\rho \in E^d$ for the given time slice t_i,
- framed in space - all points for the given slice in the given E^d space are given (limited to the (x, t) case, i.e. $x \in R^1$),
- unframed in space and time - just an unordered "heap of points" scattered in space-time.

Also, the points in the Ω domain might also be with known mutual point correspondences, i.e. a geometrical trajectory of a point can be reconstructed, e.g. using the buoys ID, or without any similar information, i.e. the buoys ID is not available. In the following, a general approach to interpolation is described and the radial basis functions (RBF) are used for interpolation.

Introduction

The Radial Basis Functions (RBF) interpolation is based on the mutual distances of points in the data domain Ω.
The RBF interpolation is given in the form:

$$h(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \varphi \|\mathbf{x} - \mathbf{x}_j\| = \sum_{j=1}^{N} \lambda_j \varphi(r_j) \tag{1}$$

where r_j is the distance from a point $\mathbf{x}$ to the point $\mathbf{x}_j$. As the parameter r of the function $\varphi(r)$ is a distance of two points in the d-dimensional space, the interpolation is non-separable by a dimension. The RBF function $\varphi(r)$ will be described in detailed later on.

Introduction

For each point, $\mathbf{x}_i$ the interpolating function has to have the value h_i. It leads to a system of linear equations:

$$h(\mathbf{x}_i) = \sum_{j=1}^{N} \lambda_j \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|) = \sum_{j=1}^{N} \lambda_j \varphi(r_{ij}) \tag{2}$$

where λ_j are unknown weights for each radial basis function, N is the number of given points and $\varphi(r)$ is the radial basis function itself.
The Eq.2 can be written if the matrix form as $\mathbf{A}\boldsymbol{\lambda} = \mathbf{h}$ or using $\varphi_{ij} = \varphi(r_{ij})$ as:

$$\begin{bmatrix} \varphi_{11} & \cdots & \varphi_{1j} & \cdots & \varphi_{1N} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \varphi_{i1} & \cdots & \varphi_{ij} & \cdots & \varphi_{iN} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \varphi_{N1} & \cdots & \varphi_{Nj} & \cdots & \varphi_{NN} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_i \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_i \\ \vdots \\ h_N \end{bmatrix} \tag{3}$$

Introduction

The interpolated value at a point $\mathbf{x}$ is computed using the Eq.1. However, due to numerical robustness and stability, additional polynomial conditions are usually added.

It can be seen, that the size of the matrix is nearly independent of the dimension and matrix size is $\approx (N \times N)$ only.

In the case of an additional polynomial $P_k(\mathbf{x}_i)$ of a degree k, we obtain:

$$h(\mathbf{x}_i) = \sum_{j=1}^{N} \lambda_j \varphi(||\mathbf{x}_i - \mathbf{x}_j||) + P_k(\mathbf{x}_i) \tag{4}$$

In the case of a bilinear polynomial $P_1(x, y)$:

$$P_1(x, y) = a_0 + a_1 x + a_2 y + a_3 xy \tag{5}$$

the additional orthogonal conditions are to be used:

$$\sum_{j=1}^{N} \lambda_j = 0 \quad \sum_{j=1}^{N} \lambda_j x_j = 0 \quad \sum_{j=1}^{N} \lambda_j y_j = 0 \quad \sum_{j=1}^{N} \lambda_j x_j y_j = 0 \tag{6}$$

Introduction

Then the RBF interpolation with the additional orthogonal conditions can be rewritten as in a more compact way:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^{\mathbf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{h} \\ \mathbf{0} \end{bmatrix} \tag{7}$$

where the matrix $\mathbf{P}$ represents the polynomial, $\boldsymbol{\lambda}$ is the vector of the RBF weights, the vector $\mathbf{a}$ contains resulting the polynomial coefficients and $\mathbf{h}$ are given values at the given points. The matrix $\mathbf{P}^{\mathbf{T}}$ represents the additional orthogonal conditions, see Eq.6. From the geometrical point of view, the polynomial $P_k(\mathbf{x})$ actually represents a rough approximation of the given data.
It should be noted that if the polynomial $P_k(\mathbf{x})$ is used:

- the RBF interpolation is not invariant to rotation and translation,
- interpolation, i.e. $\boldsymbol{\lambda}$ and $\mathbf{a}$, depends on physical units used for the vector $\mathbf{x}$,
- it might be actually counterproductive in the case of large range of domain.

It can be seen that the RBF interpolation leads to a linear system $\mathbf{Ax} = \mathbf{b}$.

Introduction

The $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ is the distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e. in the case of:

- spatial data (time independent) - usually the Euclidean norm is used:

$$r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{k=1}^{d} ({}^k x_i - {}^k x_j)^2} \tag{8}$$

  where ${}^k x_i$ means the k^{th} element of the vector $\mathbf{x}_i$,

- spatio-temporal data (time varying)

$$r_{ij} = \|(\mathbf{x}, t)_i - (\mathbf{x}, t)_j\| = \sqrt{\sum_{k=1}^{d} ({}^k x_i - {}^k x_j)^2 + \beta^2 (t_i - t_j)^2} \tag{9}$$

  where ${}^k x_i$ means the k^{th} element of the vector $\mathbf{x}_i$.
  The coefficient β has physical unit $[m/s]$ and reflects the speed of the physical phenomena, e.g. speed of sound in water, speed of light, etc.

Introduction

Another possibility is to use the $\varphi(r)$ RBF with a multiplicative exponential time term as:

$$\phi(r(t),t) = \varphi(r)\ e^{-kt} \qquad \text{or} \qquad \phi(r(t),t) = \varphi(r)\ e^{(-k_1 t^2 - k_2 t)}$$

where k, k_1 and k_2 are some positive constants, see Ku.
It should be noted, that if the points $\mathbf{x}$ are not static, i.e. $\mathbf{x} = \mathbf{x}(t)$ then the mutual distances of points are not constant, i.e. $r = r(t)$.
The above-mentioned radial basis functions are used for interpolation and approximation, solution of ordinary differential equations (ODE) and partial differential equations (PDE), etc.

Normalized RBF is another modification of the "standard" RBF. The Normalized RBF (N-RBF) is given in the form:

$$h(\mathbf{x}) = \frac{\sum_{j=1}^{N} \lambda_j \varphi \|\mathbf{x} - \mathbf{x}_j\|)}{\sum_{j=1}^{N} \varphi \|\mathbf{x} - \mathbf{x}_j\|} = \frac{\sum_{j=1}^{N} \lambda_j \varphi(r_j)}{\sum_{j=1}^{N} \varphi(r_j)} \tag{10}$$

where r_j is the distance from a point $\mathbf{x}$ to the point $\mathbf{x}_j$.
The N-RBF are used especially in the RBF neural networks applications. However, some functions $\varphi(r)$ used for the interpolation and approximation are not strictly positive, e.g. $r^2 \ln(r)$ (Thin-Plate Spline - TPS) which is negative on the interval $(0,1)$, the Euclidean norm should be used for robustness of computation, see Eq.11.

Squared Normalized RBF

The Squared Normalized RBF (SN-RBF) is given in the form:

$$h(\mathbf{x}) = \frac{\sum_{j=1}^{N} \lambda_j \varphi(\|\mathbf{x} - \mathbf{x}_j\|)}{\sqrt{\sum_{j=1}^{N} \varphi^2(\|\mathbf{x} - \mathbf{x}_j\|)}} = \frac{\sum_{j=1}^{N} \lambda_j \varphi(r_j)}{\sqrt{\sum_{j=1}^{N} \varphi^2(r_j)}} \tag{11}$$

Squared Normalized RBF

This is actually the Euclidean normalization of each row of the matrix **A** in Eq.3.

$$\sum_{j=1}^{N} \lambda_j \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|) = h(\mathbf{x}) \sqrt{\sum_{j=1}^{N} \varphi^2(r_{ij})} \quad i = 1, \ldots, N \tag{12}$$

where $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$.
It should be noted, that $O(N^2)$ division operations are replaced by $O(N)$ multiplications; the given values $h(\mathbf{x}_i)$ are just multiplied by the values $\sqrt{\sum_{j=1}^{N} \varphi^2(r_{ij})}$. This also leads to higher robustness of computation for high N.
In the case of interpolation a polynomial of a degree k $P_k(\mathbf{x})$ can be added:

$$h(\mathbf{x}) = \frac{\sum_{j=1}^{N} \lambda_j \varphi \|\mathbf{x} - \mathbf{x}_j\|)}{\sqrt{\sum_{j=1}^{N} \varphi^2(r_j)}} + P_k(\mathbf{x}) \tag{13}$$

and some orthogonal conditions have to be added as well, see Eq.6.

Squared Normalized RBF

The polynomial $P_k(\mathbf{x})$ improves the conditionality of the matrix $\mathbf{A}$ and also roughly approximate the given data. The Eq.13 can be modified similarly as the Eq.12 to:

$$\sum_{j=1}^{N} \lambda_j \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|) + P_k(\mathbf{x}_i) \sqrt{\sum_{j=1}^{N} \varphi^2(r_{ij})} = h(\mathbf{x}_i) \sqrt{\sum_{j=1}^{N} \varphi^2(r_{ij})} \quad i = 1, \ldots, N \quad (14)$$

As the $q_i = \sqrt{\sum_{j=1}^{N} \varphi^2(r_{ij})}$ is constant for the i^{th} row ($i = 1, \ldots, N$) in the Eq.3, the Eq.14 can be simplified to:

$$\sum_{j=1}^{N} \lambda_j \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|) + q_i \; P_k(\mathbf{x}_i) = q_i \; h(\mathbf{x}_i) \quad i = 1, \ldots, N \quad (15)$$

Squared Normalized RBF

The matrix for the SN-RBF interpolation has form:

$$\begin{bmatrix} \varphi_{11} & \dots & \varphi_{1N} & q_1 & q_1x_1 & q_1y_1 & q_1x_1y_1 \\ \vdots & \ddots & \vdots & 1 & \vdots & \vdots & \vdots \\ \varphi_{N1} & \dots & \varphi_{NN} & q_N & q_Nx_N & q_Ny_N & q_Nx_ny_N \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ x_1 & \dots & x_N & 0 & 0 & 0 & 0 \\ y_1 & \dots & y_N & 0 & 0 & 0 & 0 \\ x_1y_1 & \dots & x_Ny_N & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_1h_1 \\ \vdots \\ q_Nh_N \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{16}$$

$$\begin{bmatrix} \mathbf{A} & \mathbf{QP} \\ \mathbf{P^T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{Qh} \\ \mathbf{0} \end{bmatrix} \tag{17}$$

where $\mathbf{Q} = diag[q_1, \dots, q_N]$ is a diagonal matrix and $q_i > 0$, $i = 1, \dots, N$. However, in the approximation case, i.e. the matrix $\mathbf{A}$ is $(N \times M), N > M$, the polynomial part has to be handled differently and the Least Square Method (LSE) cannot be used directly.

Squared Normalized RBF Functions

There are several radial basis functions and can be divided into two major groups:

- "global" RBFs having global influence, e.g.
  - Polyharmonic spline:
    $$\varphi(r) = r^k, \quad k = 1,3,5,\ldots$$
    $$\varphi(r) = r^k \ln r, \quad k = 2,4,6,\ldots$$
  - Thin plate spline [TPS](a special polyharmonic spline): $\varphi(r) = r^2 \ln r$,
  - Gaussian: $\varphi(r) = e^{-\alpha r^2}$
  - Multiquadric: $\sqrt{1+\alpha r^2}$
  - Inverse quadratic: $\frac{1}{1+\alpha r^2}$
  - Inverse multiquadratic: $\frac{1}{\sqrt{1+\alpha r^2}}$

  where $\alpha > 0$ is a shape parameter, $r \in < 0, \infty)$.
  The RBF matrix is usually full and the matrix can be very ill conditioned.
- "local" RBFs - Compactly Supported RBF (CS-RBF) have a non-zero positive value on the interval $< 0,1)$ only. The RBF matrix is usually sparse and it depends on the shape parameter α.

Compactly supported RBF

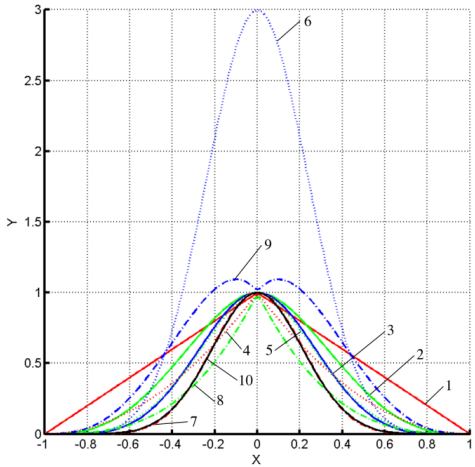


Figure: Compactly Supported RBF - CSRBF

Squared Normalized RBF

There are other RBFs used, e.g. the "bumb CS-RBF" based on Gaussian:

$$\varphi(r) = \begin{cases} e^{-\frac{1}{1-r^2}} & r \in <0,1) \\ 0 & r \geq 1 \end{cases}$$

Buhmann proposed CS-RBF in the form:

$$\varphi(r) = \begin{cases} \frac{1}{3} + r^2 - \frac{4}{3}r^3 + 2r^2 \log r & r \in <0,1) \\ 0 & r \geq 1 \end{cases}$$

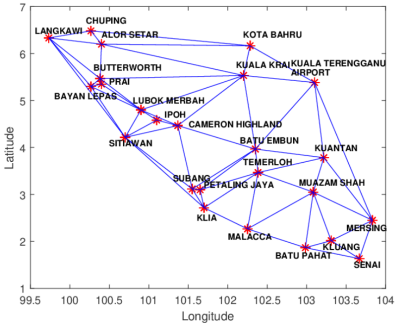
and Manandro proposed two new rational CS-RBF classes.

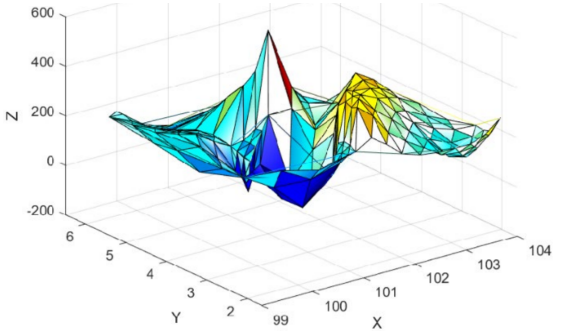
Squared Normalized RBF

RBF interpolation example For demonstration of the RBF spatio-temporal scattered data interpolation properties, the Rainfall data of the Peninsular Malaysia from the Malaysian Meteorology Department in 2007 were taken. The data are static from the spatial point of view and dynamic from the temporal one. One possible approach is to make the domain tessellation, then subdivide the mesh and then smooth it.

Squared Normalized RBF

Ali used Delaunay triangulation on 25 major meteorological stations and used cubic Timmer triangular patches for the surface representation and interpolation.

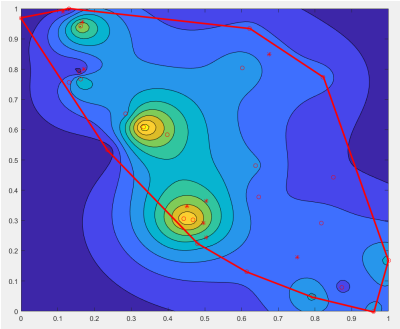




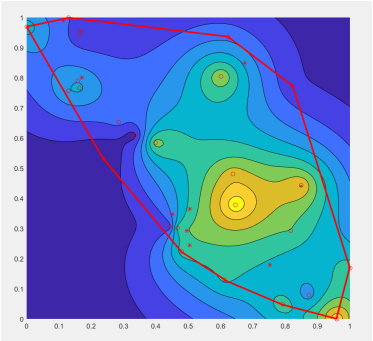
It should be noticed, that additional operations are need in order to obtain a smooth surface over the triangular mesh generated by the Ali's algorithm.

Experimental Results

In the following, a solution based on the RBF interpolation is presented.



March 2007



May 2007

The RBF approach leads to smooth surface automatically, however, interpolation on borders might not be reliable.

Experimental Results

The 3D views of the Rainfall interpolated data are presented in Fig.4 and Fig.5. It should be noted that the α shape parameter has to be set *reasonably*.

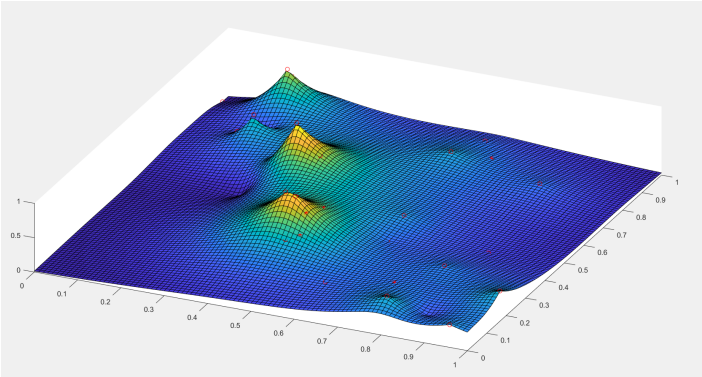


Figure: Interpolation of the Rainfall data March 2007
the Gauss function used, shape parameter $\alpha = 0.255$

Experimental Results

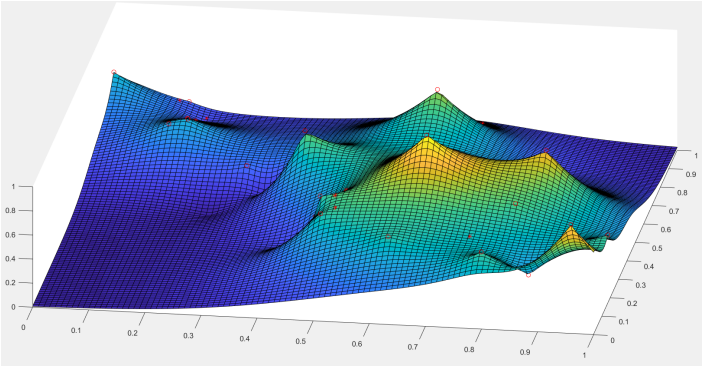


Figure: Interpolation of the Rainfall data May 2007
the Gauss function used, shape parameter $\alpha = 0.2550$

Squared Normalized RBF

In general, the RBF interpolation benefits from the following properties:

- the final interpolation is naturally smooth,
- if CS-RBFs are used, the RBF matrix is sparse,
- computational complexity is nearly independent of the dimensionality, depends on the number of points,
- an explicit formal analytical formula of the final interpolation is obtained,
- if the RBF function $\varphi(r)$ is positive definite, iterative methods for solving linear system of equations can be used,
- the SN-RBF increases the numerical stability as it actually normalizes each row of the RBF matrix,
- as a solution of linear system of equations is equivalent to the outer product (extended cross product) use, the standard symbolic operations can be used for further processing *without need of the numerical evaluation*,
- the block matrix decomposition might be used in the case of large data,
- the domain decomposition can be applied, which leads to faster computation.

Conclusions

This contribution briefly presents a new form of normalized RBF, the Squared Normalized RBF SN-RBF, for the spatio-temporal data and a perspective of use in sensor networks, when sensors do not have a fixed position and data are not synchronized in time, i.e. in the time-slots. The proposed SN-RBF formulation leads to better RBF matrix conditionality, too.

The presented SN-RBF interpolation method is especially convenient for scattered spatio-temporal interpolation of data cases, e.g. when the sensors are transmitting data only in the case, when physical phenomena reach values outside the expected range and/or changing their positions, e.g. surveillance sensors in the sea, etc.

Future work is to be targeted to analysis of the β parameter setting and its sensitivity, to the TPS $(r^2 \ln r)$ function applicability for large data sets, conditionality of the resulting RBF matrices and to methods for visualization of spatio-temporal data for 2D+T and 3D+T data as they cannot be visualized using methods like contour plots or 3D projections.

Acknowledgments

The author thanks to colleagues and colleagues at the Shandong University(Jinan) and Zhejiang University(Hangzhou) China, University of West Bohemia, Pilsen for their critical comments, discussions, especially to colleagues Martin Cervenka, Mariia Martynova and Jan Kasak for producing some images and for counter numerical verifications.

REFERENCES

Please, see the proceedings

Questions ?