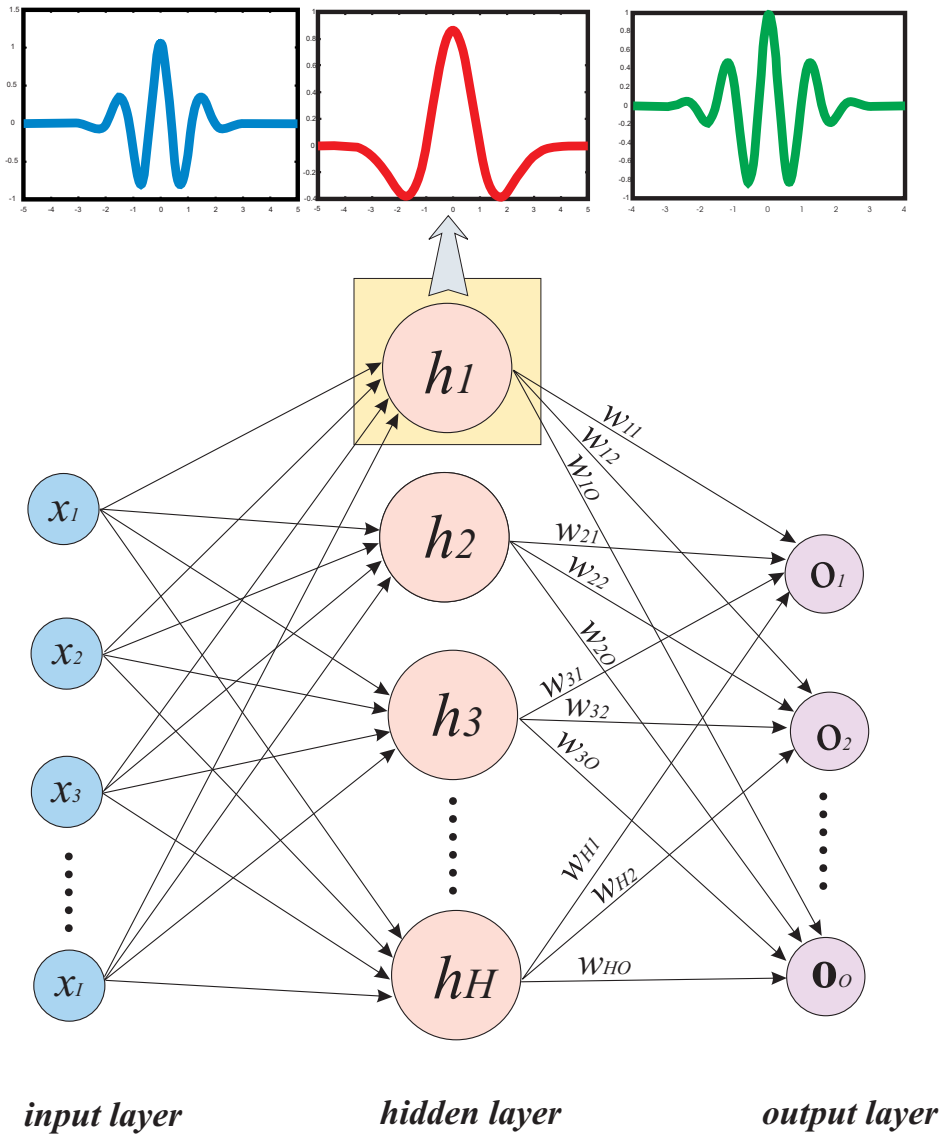
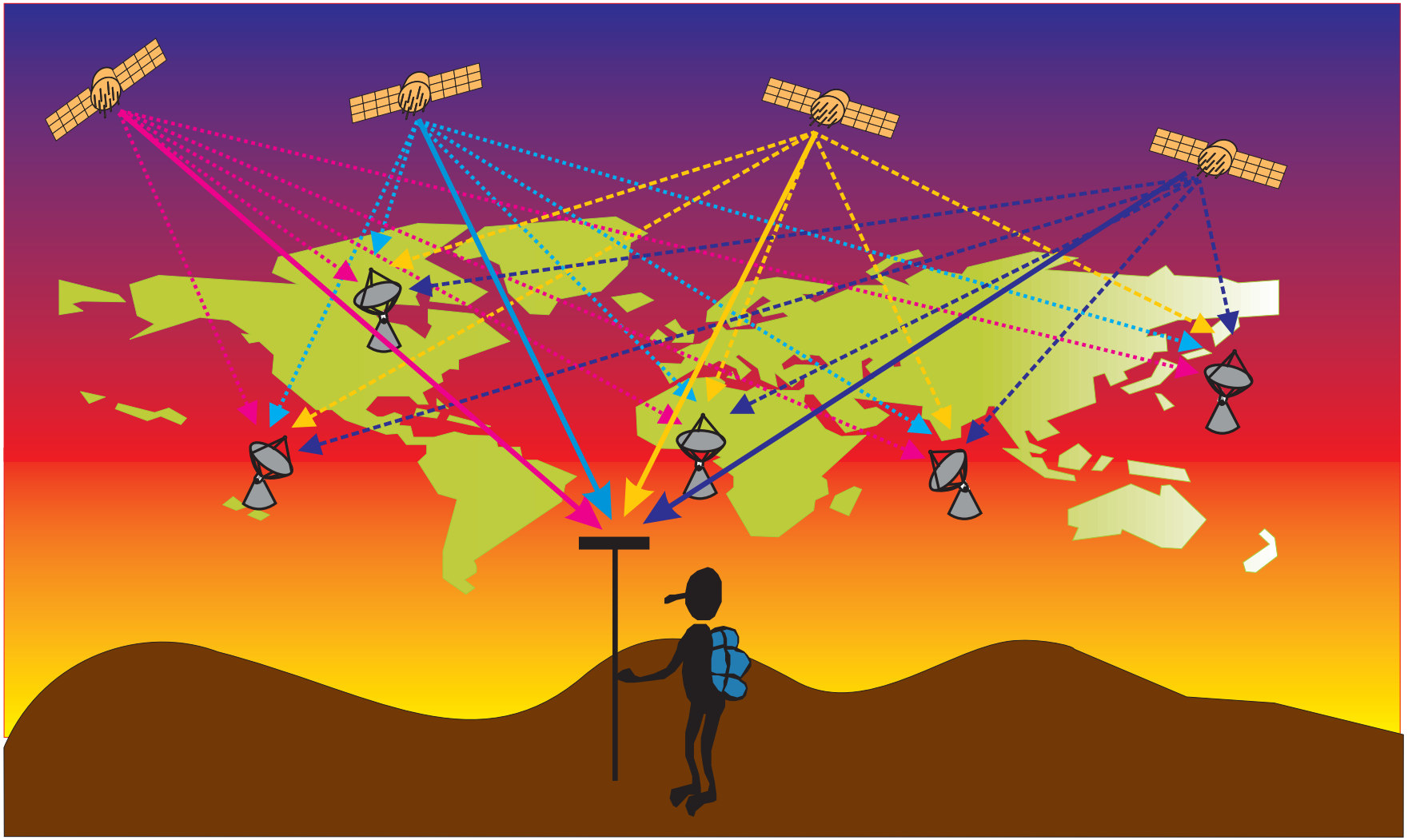
Wavelet neural networks in function approximation

P. Pavlovčič Prešeren, B. Stopar, O. Sterle
University of Ljubljana, Slovenia

1^{st} International Conference on new advanced GNSS and 3D spatial technologies: APPLICATIONS IN CIVIL AND ENVIRONMENTAL ENGINEERING, GEOPHYSICS, ARCHEOLOGY AND CULTURAL HERITAGE
TRIESTE, 18–20 February 2016

Figure :
http://www.dreamstime.com/photos-images/brain.html


h1
h2
h3
hH
x1
x2
x3
xI
o1
o2
oO
w11
w12
w1O
w21
w22
w2O
w31
w32
w3O
wH1
wH2
wHO
input layer
hidden layer
output layer



Contents

Function approximation

Formalization:

- we have a finite set od data points, i. e. inputs: **x** and outputs **y**
- we don't know the functional form $\mathbf{f} : \mathbf{X} \rightarrow \mathbf{Y}$ (ideal formula)

Function approximation

Formalization:

- we have a finite set od data points, i. e. inputs: $\mathbf{x}$ and outputs $\mathbf{y}$
- we don't know the functional form $\mathbf{f} : \mathbf{X} \rightarrow \mathbf{Y}$ (ideal formula)
- we want to determine the underlying functional form from data, based on specific assumption:

$$(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$$

$$\downarrow \quad \downarrow \qquad \downarrow$$

Function approximation

Formalization:

- we have a finite set od data points, i. e. inputs: $\mathbf{x}$ and outputs $\mathbf{y}$
- we don't know the functional form $\mathbf{f} : \mathbf{X} \rightarrow \mathbf{Y}$ (ideal formula)
- we want to determine the underlying functional form from data, based on specific assumption:

$$(\mathbf{x_1}, \mathbf{y_1}), (\mathbf{x_2}, \mathbf{y_2}), \ldots, (\mathbf{x_N}, \mathbf{y_N})$$

$$\downarrow \quad \downarrow \qquad \downarrow$$

- final result is the hypothesis: $\mathbf{g} : \mathbf{X} \rightarrow \mathbf{Y}$ (**formula to be used**)

Function approximation

Formalization:

- we have a finite set od data points, i. e. inputs: $\mathbf{x}$ and outputs $\mathbf{y}$
- we don't know the functional form $\mathbf{f} : \mathbf{X} \rightarrow \mathbf{Y}$ (ideal formula)
- we want to determine the underlying functional form from data, based on specific assumption:

$$(\mathbf{x_1}, \mathbf{y_1}), (\mathbf{x_2}, \mathbf{y_2}), ..., (\mathbf{x_N}, \mathbf{y_N})$$

$$\downarrow \quad \downarrow \qquad \downarrow$$

- final result is the hypothesis: $\mathbf{g} : \mathbf{X} \rightarrow \mathbf{Y}$ (**formula to be used**)

Polynomial interpolation

Lagrange interpolation is very often used in SP3 orbit interpolation (equally spaced tabular data).

Coefficients of the interpolation polynomial are given by:

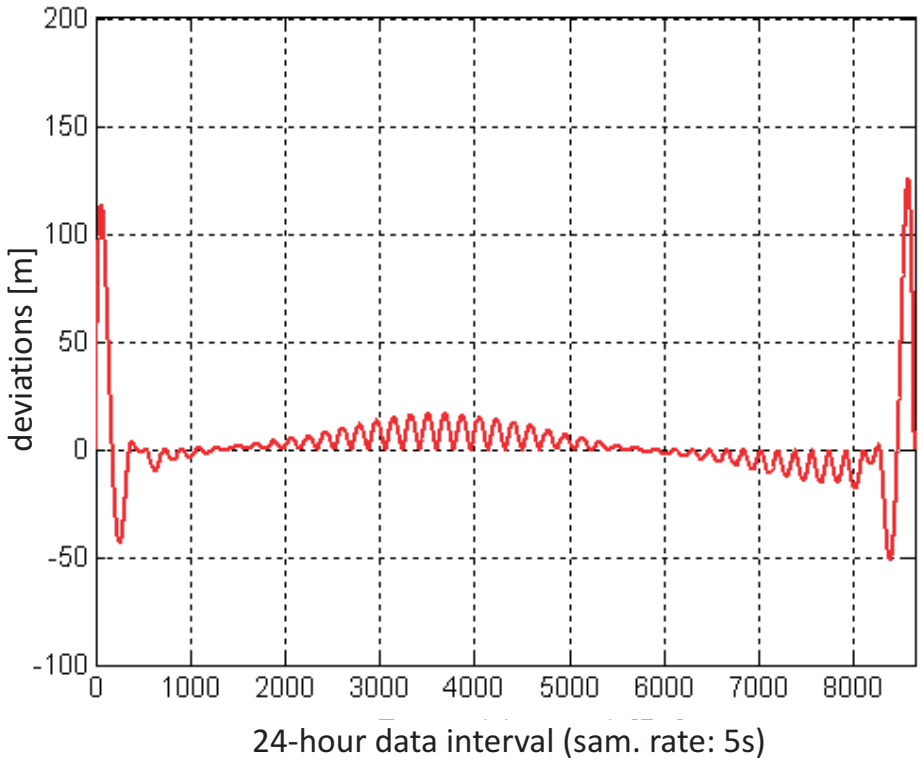
$$l_i(t) = \frac{\prod_{k=0}^{i-1}(t - t_k)}{\prod_{k=0}^{i-1}(t_i - t_k)} \cdot \frac{\prod_{k=i+1}^{n}(t - t_k)}{\prod_{k=i+1}^{i-1}(t_i - t_k)} \tag{1}$$

The interpolation polynomial is given by:

$$p_n(t) = \sum_{i=0}^{n} f(t_i) l_i(t) \tag{2}$$

Polynomial interpolation

Problems of polynomial interpolation: oscillation

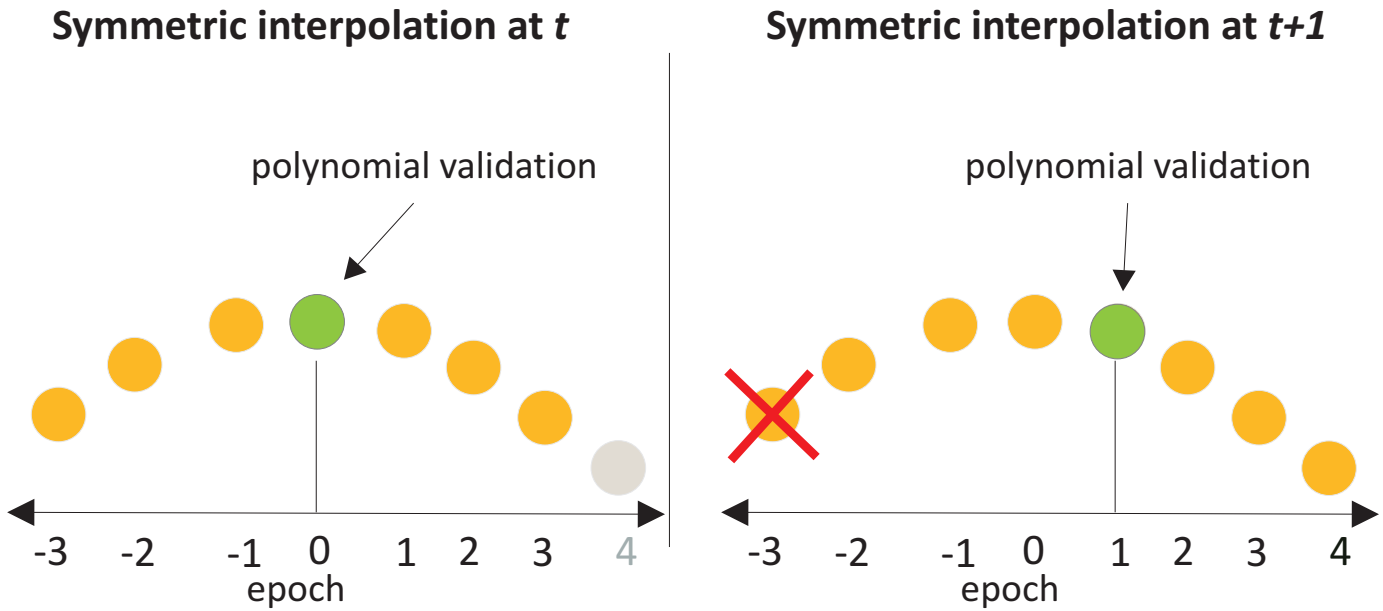


Calculation accuracy is not good enough at the beginning/end of the interval.

Successive interpolations

The solution of the problem:

– providing more functions for definition area (successive interpolations).



Artificial neural networks

- are based on learning/mapping the input **x** to output **y** data
- we acquire the model to compute function values elsewhere.

Artificial neural networks

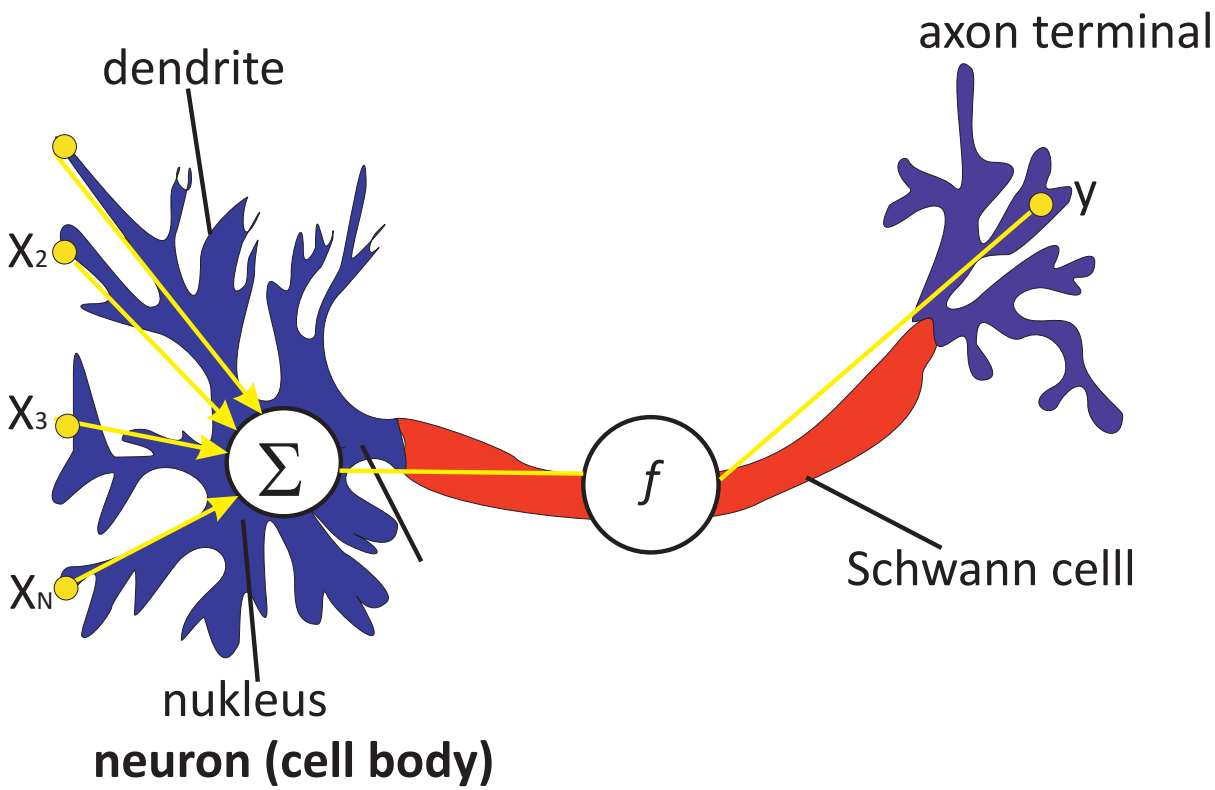
- are based on learning/mapping the input **x** to output **y** data
- we acquire the model to compute function values elsewhere.

biological function $\longrightarrow$ biological structure

Artificial neural networks

- are based on learning/mapping the input **x** to output **y** data
- we acquire the model to compute function values elsewhere.

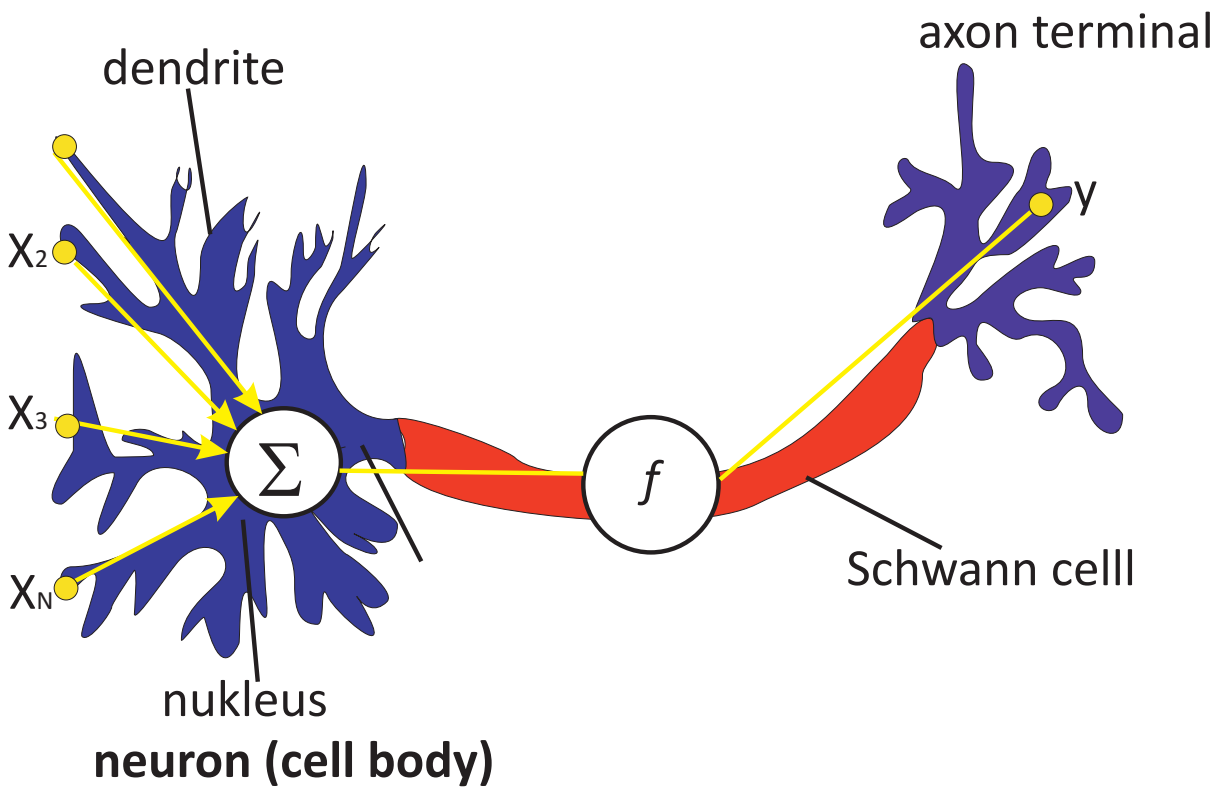
biological function $\longrightarrow$ biological structure



Artificial neural networks

- are based on learning/mapping the input **x** to output **y** data
- we acquire the model to compute function values elsewhere.

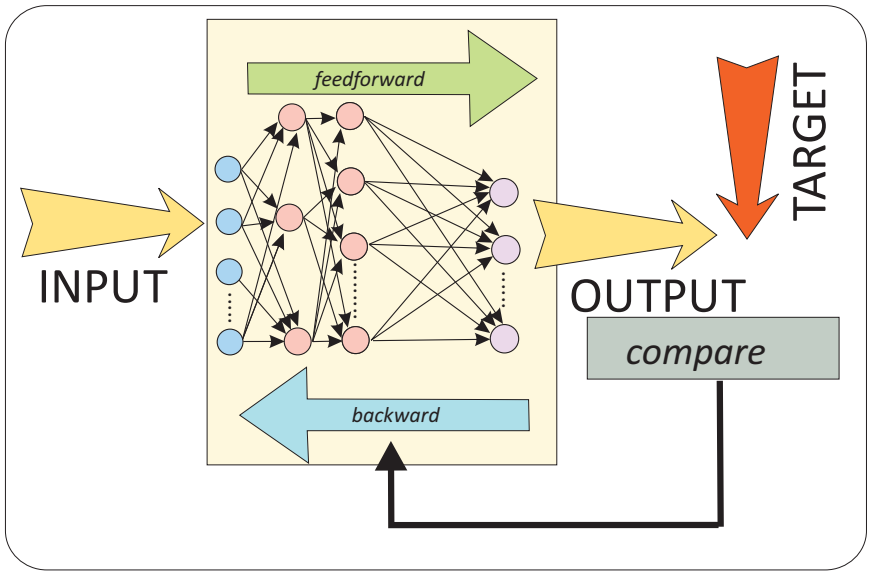
biological function $\longrightarrow$ biological structure



Neural network model

1. input layer (data)
2. hidden layers
3. output layer (data)

Connections are the weights; each neuron has its activation function.



Neural network learning – backpropagation

The algorithm can be used with different activation functions (linear, sigmoid,...)

1. weights w_{ij} are randomly initialized
2. forward phase: computation of the target values

Neural network learning – backpropagation

The algorithm can be used with different activation functions (linear, sigmoid,...)

1. weights w_{ij} are randomly initialized
2. forward phase: computation of the target values
3. at the end: computation of the errors δ

Neural network learning – backpropagation

The algorithm can be used with different activation functions (linear, sigmoid,...)

1. weights w_{ij} are randomly initialized
2. forward phase: computation of the target values
3. at the end: computation of the errors δ
4. backward phase: updating all the weights

Neural network learning – backpropagation

The algorithm can be used with different activation functions (linear, sigmoid,...)

1. weights w_{ij} are randomly initialized
2. forward phase: computation of the target values
3. at the end: computation of the errors δ
4. backward phase: updating all the weights

Algorithm is time consuming and can be trapped into a local instead of a global minima.

Neural network learning – backpropagation

The algorithm can be used with different activation functions (linear, sigmoid,...)

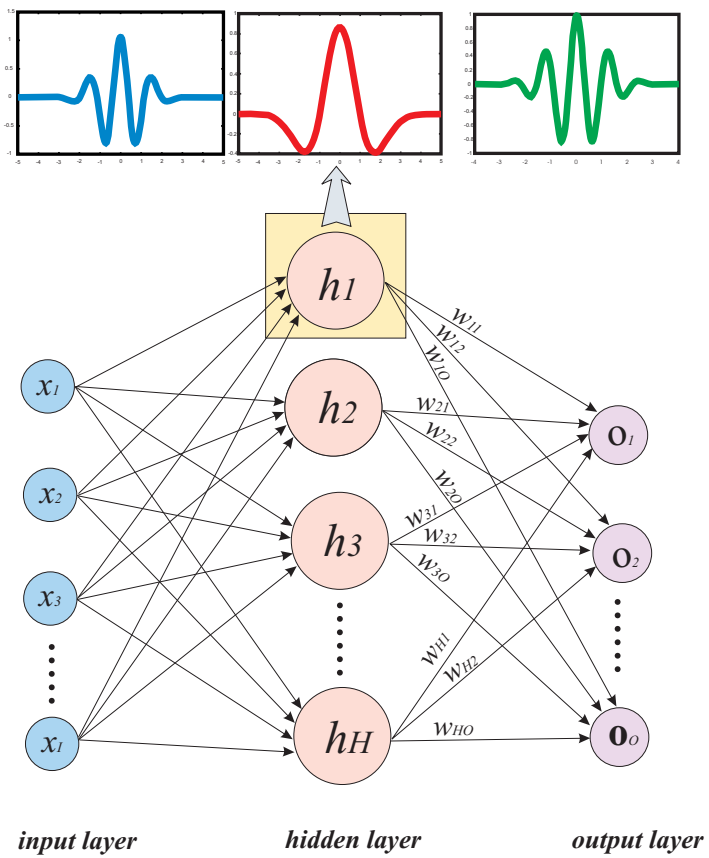
1. weights w_{ij} are randomly initialized
2. forward phase: computation of the target values
3. at the end: computation of the errors δ
4. backward phase: updating all the weights

Algorithm is time consuming and can be trapped into a local instead of a global minima.

Wavelet neural networks

WNNs are a special kind of ANNs. They combine two theories:

- **wavelet theory** (optimization of theorems of approximation/scaling)
- **feedforward networks** (universal approximation property)



Wavelets

Different functions could be chosen for mother wavelet:

- Mexican hat: $\psi(x) = (1 - 2x^2)\exp(-x^2)$
- Morlet: $\psi(x) = \cos(5x)\exp(-x^2/2)$

Wavelets

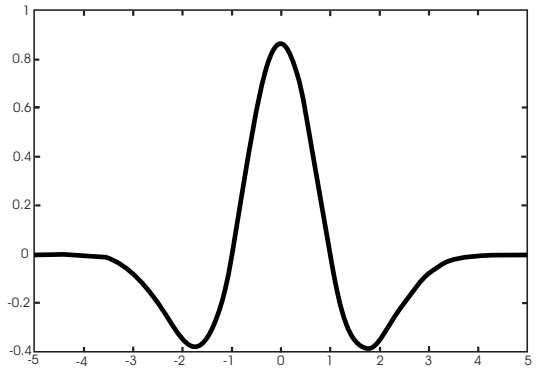
Different functions could be chosen for mother wavelet:

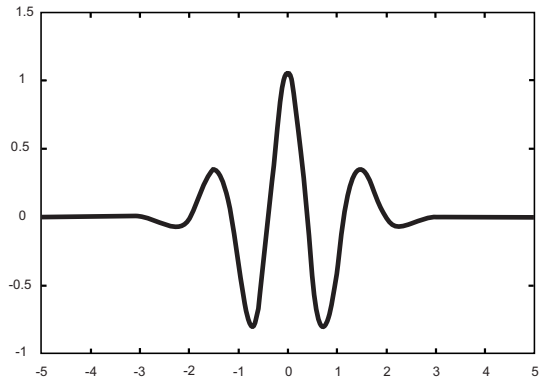
- Mexican hat: $\psi(x) = (1 - 2x^2)\exp(-x^2)$
- Morlet: $\psi(x) = \cos(5x)\exp(-x^2/2)$
- Gaussian: $\psi(x) = (-x)\exp(-x^2/2)$

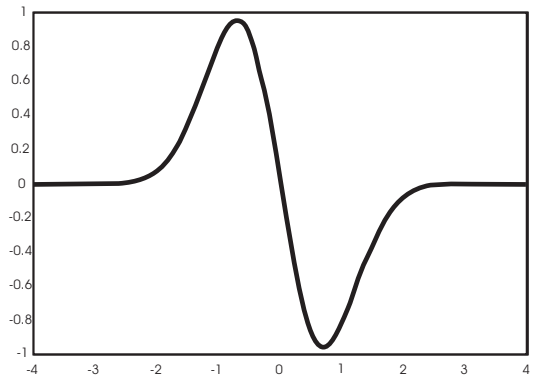
Wavelets

Different functions could be chosen for mother wavelet:

- Mexican hat: $\psi(x) = (1 - 2x^2)\exp(-x^2)$
- Morlet: $\psi(x) = \cos(5x)\exp(-x^2/2)$
- Gaussian: $\psi(x) = (-x)\exp(-x^2/2)$



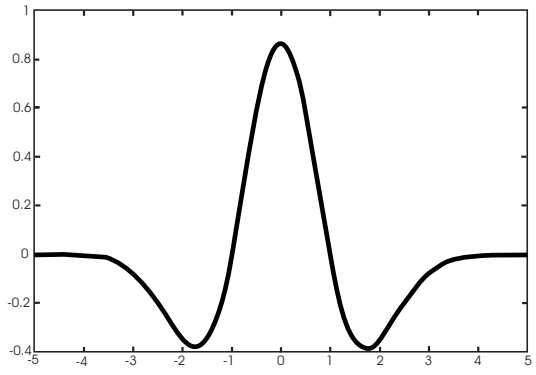


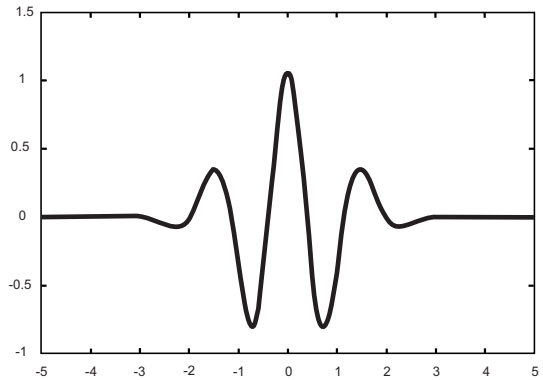


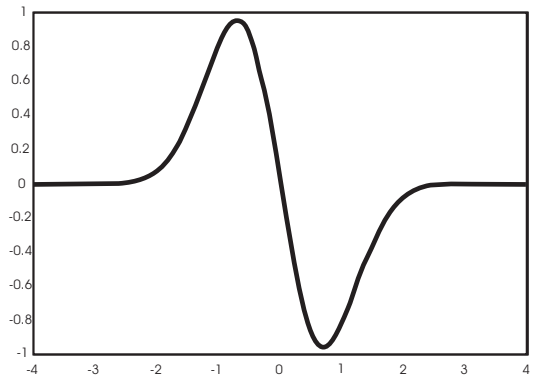
Wavelets

Different functions could be chosen for mother wavelet:

- Mexican hat: $\psi(x) = (1 - 2x^2)\exp(-x^2)$
- Morlet: $\psi(x) = \cos(5x)\exp(-x^2/2)$
- Gaussian: $\psi(x) = (-x)\exp(-x^2/2)$



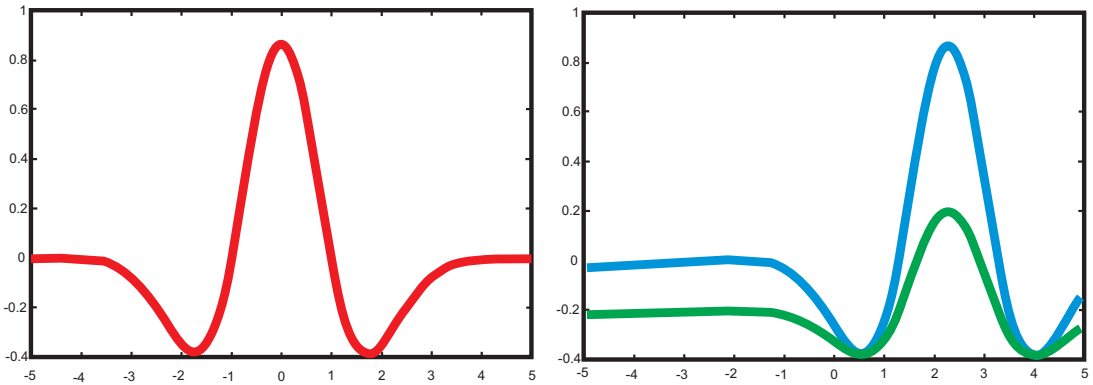




Daughter wavelets

Daughter wavelets:

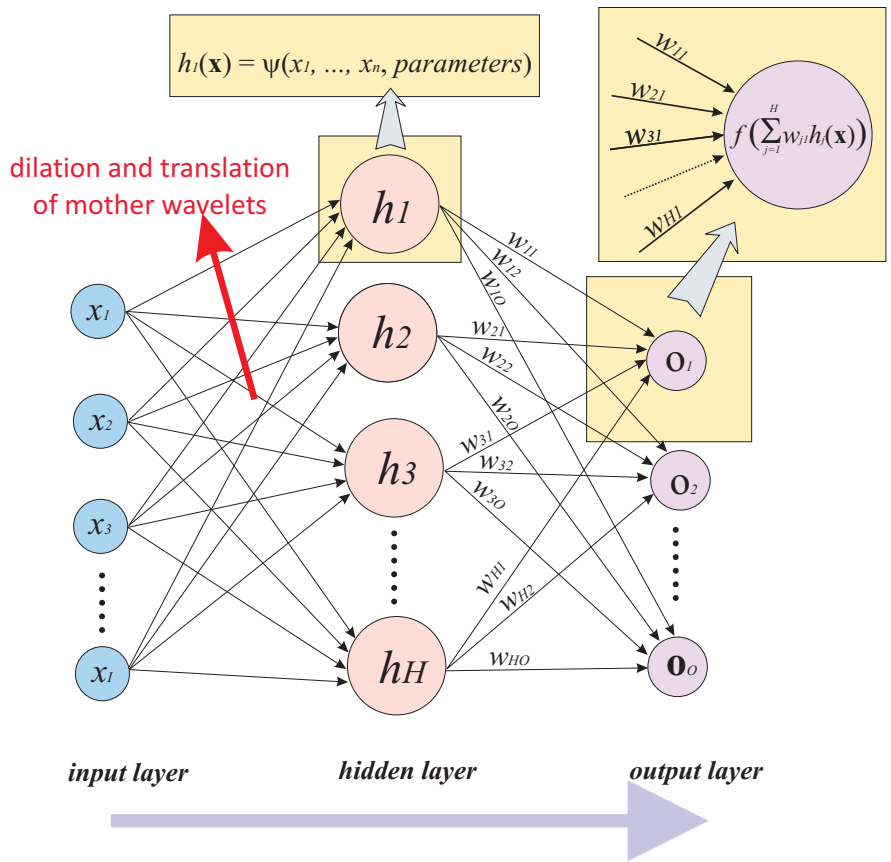
$$\psi_j(\mathbf{x}) = a_j^{-d/2}\psi\left(\frac{\mathbf{x}-b_j}{a_j}\right)$$



Two steps of learning algorithm in WNNs:

- finding and fixing dilation and translation parameters
- finding the weights: $w_1, w_2, ..., w_n$

WNN model



WNN output is given by:

$$\mathbf{f} = \sum_{j=1}^{m} \mathbf{w}_j \psi_j(\mathbf{x})$$

WNN learning

Since WNNs are based on:

- a single hidden layer unit and
- all basis functions are orthogonal

we follow a different algorithm for finding the weights.

Our goal is to find the optimal WNN structure (number of hidden layer neurons), which minimizes the error cost function:

$$E(f) = \frac{1}{2} \sum_{i=1}^{N} (y_i - o_i)^2$$

WNN finding the weights

The network output $\mathbf{O}$ for all the input data data can be expressed as:

$$\mathbf{O} = \boldsymbol{\Psi} \cdot \mathbf{W}$$

where:

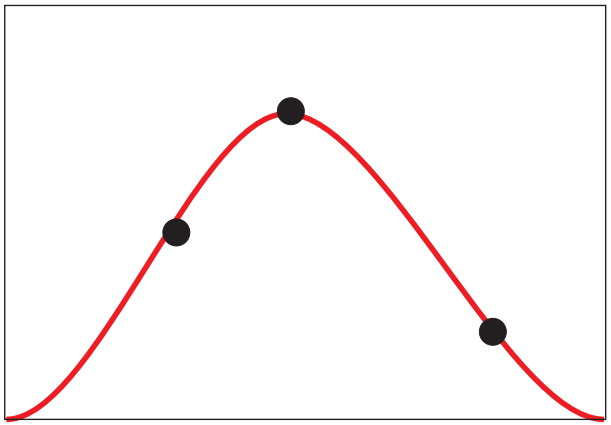
$$\boldsymbol{\Psi} = \begin{bmatrix} \psi(x_1, a_1, b_1) & \cdots & \psi(x_1, a_n, b_n) \\ \psi(x_2, a_1, b_1) & \cdots & \psi(x_2, a_n, b_n) \\ \cdots & \cdots & \cdots \\ \psi(x_n, a_1, b_1) & \cdots & \psi(x_n, a_n, b_n) \end{bmatrix}$$

The weight matrix is solved by pseudoinversion:

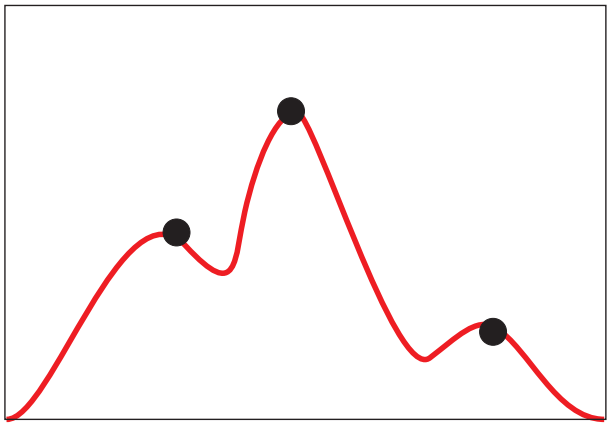
$$\mathbf{W} = \boldsymbol{\Psi}^{+}\mathbf{O} \qquad \boldsymbol{\Psi}^{+} = \left(\boldsymbol{\Psi}^{T}\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^{T}$$

Choosing wavelet parameters

Dilation and translation parameters should be properly chosen:



large dilation



small dilation

For each new neuron the procedure of optimal model follows iterative steps:

1. we choose dilation/translation according to the input data and fix parameters to solve the weights
2. we fix weights to minimize error function (for the dilation).

Case study: SP3 orbit approximation

- we have used 9 training tabular data (15 minut interval) $\rightarrow$ from SP3 files
- 25 5-minute testing data $\rightarrow$ acquired from numerical integration (SP3 position and velocity data)

Table : Statistics for radial component (Δr) (9 training and 25 testing data)

hidden layer function	δ_{min}	δ_{max}	$\bar{\delta}$
Mexican hat	0.0000 m	0.0052 m	0.0017 m
Morlet	0.0011 m	0.0071 m	0.0044 m
Gaussian	0.0000 m	0.0052 m	0.0016 m
RBF	0.0009 m	0.0078 m	0.0015 m

Table : Statistics for x, y and z component

hidden layer function	δ_{min}	δ_{max}	$\bar{\delta}$
Mexican hat	-0.0050 m	0.0035 m	$4.81 \cdot 10^{-5}$ m
Morlet	-0.0051 m	0.0031 m	$6.40 \cdot 10^{-5}$ m
Gaussian	-0.0050 m	0.0035 m	$4.90 \cdot 10^{-5}$ m
RBF	-0.0058 m	0.0039 m	$2.11 \cdot 10^{-4}$ m

hidden layer function	δ_{min}	δ_{max}	$\bar{\delta}$
Mexican hat	-0.0006 m	0.0008 m	$1.14 \cdot 10^{-5}$ m
Morlet	-0.0007 m	0.0000 m	$3.18 \cdot 10^{-5}$ m
Gaussian	-0.0006 m	0.0001 m	$1.13 \cdot 10^{-4}$ m
RBF	-0.0039 m	0.0042 m	$7.42 \cdot 10^{-5}$ m

hidden layer function	δ_{min}	δ_{max}	$\bar{\delta}$
Mexican hat	-0.0015 m	0.0012 m	$-1.20 \cdot 10^{-4}$ m
Morlet	-0.0058 m	0.0071 m	$1.48 \cdot 10^{-6}$ m 9
Gaussian	-0.0015 m	0.0012 m	$-1.20 \cdot 10^{-4}$ m
RBF	-0.0082 m	0.0098 m	$-0.87 \cdot 10^{-6}$ m

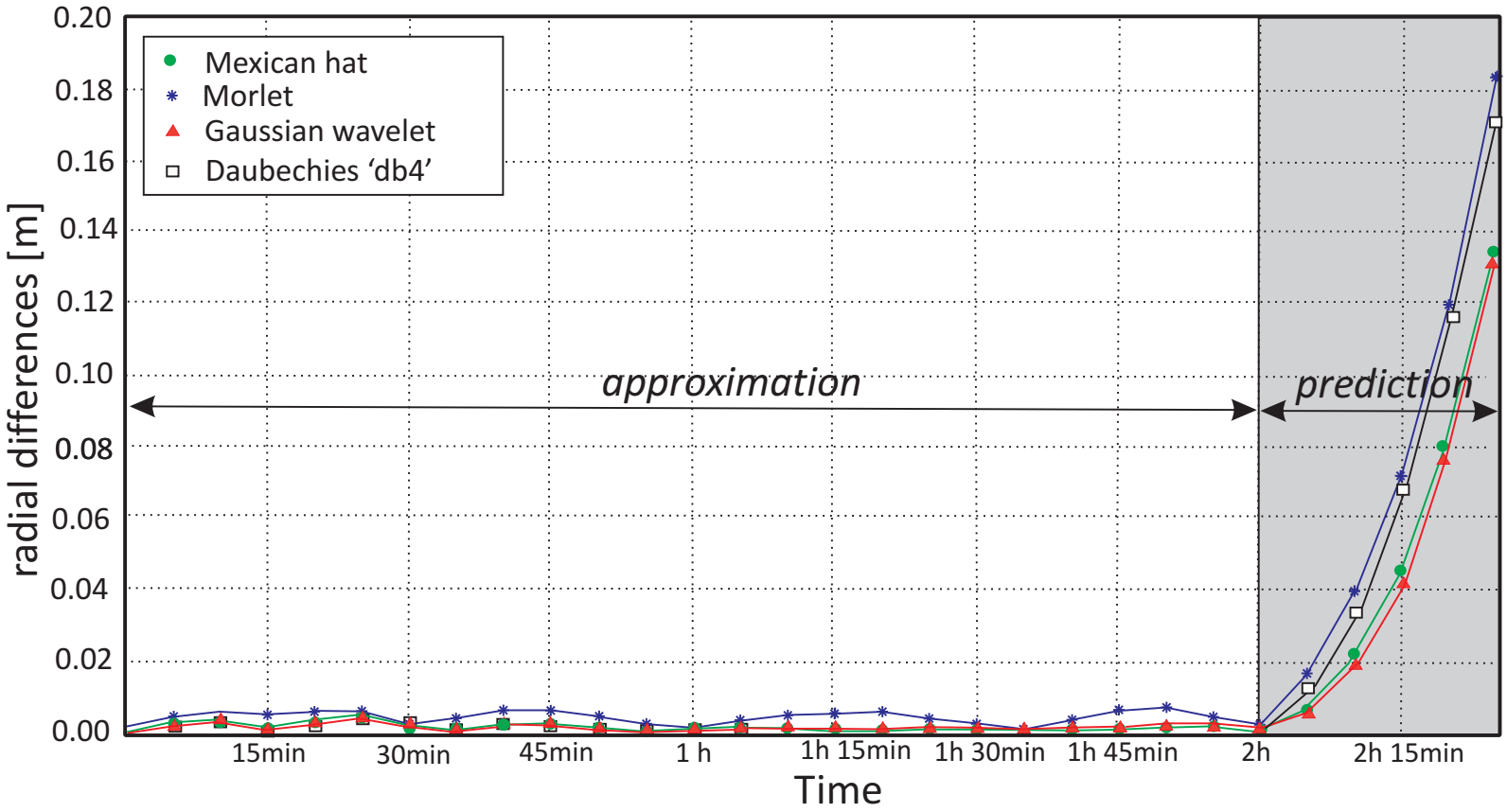


Figure : Different WNNs approximation and prediction performance for GNSS orbit (radial component Δr).

Summary

- we have shown a different way to provide unique continuous orbit functions for GNSS satellites
- the computation method is based on the WNN optimal model estimation

Summary

- we have shown a different way to provide unique continuous orbit functions for GNSS satellites
- the computation method is based on the WNN optimal model estimation
- WNNs are appropriate for continuous orbit function construction (smaller oscillations)

Summary

- we have shown a different way to provide unique continuous orbit functions for GNSS satellites
- the computation method is based on the WNN optimal model estimation
- WNNs are appropriate for continuous orbit function construction (smaller oscillations)
- WNNs could be used also for short-time prediction (if no broadcast/precise efemerides are available)

Summary

- we have shown a different way to provide unique continuous orbit functions for GNSS satellites
- the computation method is based on the WNN optimal model estimation
- WNNs are appropriate for continuous orbit function construction (smaller oscillations)
- WNNs could be used also for short-time prediction (if no broadcast/precise efemerides are available)

THANK YOU!