# Difference between ODE-Net approximation and ODEsolvers: Euler, RK4.

## Содержание

# 1 Report

## 1.1 Motivation

Many methods have been developed so far for solving differential equations. Some of them produce a solution in the form of an array that contains the value of the solution at a selected group of points. Others use basis-functions to represent the solution in analytic form and transform the original problem usually in a system of linear equations. Another approach to the solution of ordinary differential equations is based on the fact that certain types of splines, for instance linear B-splines, can be derived by the superposition of piecewise linear activation functions. ODE-Net view the problem from a different angle. It relies on the function approximation capabilities of feedforward neural networks and results in the construction of a solution written in a diferentiable, closed analytic form.

## 1.2 Description of the method

General differential equation definition:

$$G(\vec{\boldsymbol{x}}, \boldsymbol{\Psi}(\vec{\boldsymbol{x}}), \nabla\boldsymbol{\Psi}(\vec{\boldsymbol{x}}), \nabla^2\boldsymbol{\Psi}(\vec{\boldsymbol{x}})) = 0, \quad \vec{\boldsymbol{x}} \in D$$

Assumes a discretization of the domain $D$

$$G(\vec{\boldsymbol{x_i}}, \boldsymbol{\Psi}(\vec{\boldsymbol{x_i}}), \nabla\boldsymbol{\Psi}(\vec{\boldsymbol{x_i}}), \nabla^2\boldsymbol{\Psi}(\vec{\boldsymbol{x_i}})) = 0, \quad \vec{\boldsymbol{x_i}} \in \hat{D}$$

With adjustable parameters $\vec{\boldsymbol{p}}$, the problem is transformed to

$$\min_{\vec{\boldsymbol{p}}} {}_{\vec{\boldsymbol{x_i}} \in \hat{D}} G(\vec{\boldsymbol{x_i}}, \boldsymbol{\Psi}_t(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{p}}), \nabla\boldsymbol{\Psi}(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{p}}), \nabla^2\boldsymbol{\Psi}(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{p}}))$$

The efficient minimization of equation can be considered as a procedure of training the neural network where the error corresponding to each input vector $\vec{\boldsymbol{x_i}}$ is the value $G(\vec{\boldsymbol{x_i}})$ which has to become zero.
In the proposed approach the trial solution $\boldsymbol{\Psi}_t$ employs a feedforward neural network and the parameters $\vec{\boldsymbol{p}}$ correspond to the weights and biases of the neural architecture. We choose a form for the trial function $\boldsymbol{\Psi}_t(\vec{\boldsymbol{x_i}})$ such that by construction satisfies the certain boundary conditions (B.Cs)

$$\boldsymbol{\Psi}_t(\vec{\boldsymbol{x_i}}) = A(\vec{\boldsymbol{x_i}}) + F(\vec{\boldsymbol{x_i}}, N(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{p}}))$$

where $N(\vec{\boldsymbol{x_i}}, \vec{\boldsymbol{p}})$ is a single-output feedforward neural network with parameters $\vec{\boldsymbol{p}}$ and $n$ input units fed with the input vector $\vec{\boldsymbol{x}}$.

## 1.3 Experiment

The code of the experiment can be found on GitHub repository[1].

[1]https://github.com/Tonchik-hv/Math-methods-of-forecasting

## 1.4 Code analysis

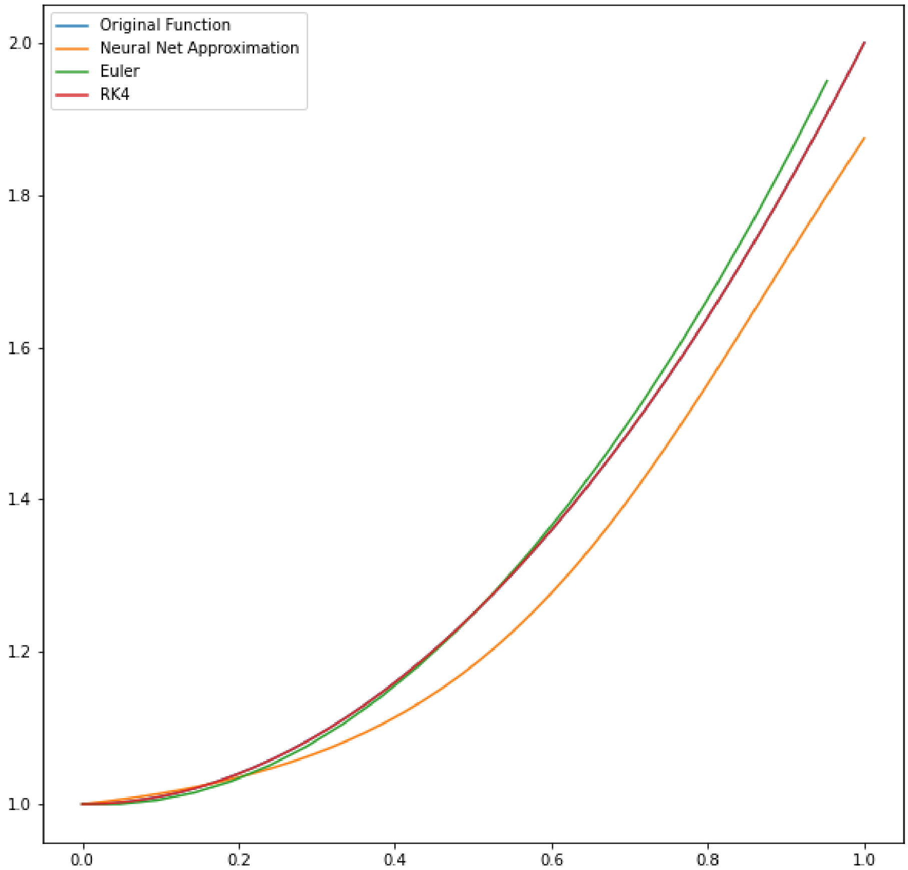


Рис. 1: Selected HOSVD mode structures for the radial (x) coordinate

## 1.5 Conclusions

1. The solution via ANN's is easily used in any subsequent calculation.
2. Most other techniques offer a discrete solution (for example Runge-Kutta methods)
3. The required number of model parameters is far less than any other solution technique and therefore, compact solution models are obtained, with very low demand on memory space.
4. The method is general and can be applied to ODEs, systems of ODEs and to PDEs as well.
5. The method can also be efficiently implemented on parallel architectures.