De-xuan Zou, Li-qun Gao, Steven Li, 2014. Volterra filter modeling of a nonlinear discrete-time system based on a ranked differential evolution algorithm. *Journal of Zhejiang University-SCIENCE C (Computers & Electronics)*, **15**(8):687-696. [doi:10.1631/jzus.C1300350]

Volterra filter modeling of a nonlinear discrete-time system based on a ranked differential evolution algorithm

Key words: Ranked differential evolution, Identification problem, Nonlinear discrete-time systems, Volterra filter model, Premature convergence

Corresponding author: De-xuan Zou
E-mail: zoudexuan@163.com

Motivation

- ➢ Disadvantages of existing methods:
  - ✓ The existing algorithms to update Volterra kernel vector ***H*** consist of an exact Newton algorithm, least mean squares algorithm, and recursive least squares algorithm, etc.
  - ✓ These algorithms are virtually based on the gradient method. The gradient method is easy to trap at a local optimum, not a global one.
  - ✓ The convergence of this method is slow.

- A ranked differential evolution (RDE) algorithm is used to solve the identification problem of nonlinear discrete-time systems based on a Volterra filter model.

Features of our method

- This paper presents a ranked differential evolution algorithm (RDE) for solving the identification problem of nonlinear discrete-time systems based on the Volterra filter model
- RDE generates a scale factor by combining a sine function and randomness
- RDE modifies mutation operation after ranking all candidate solutions of the population
- Preserve a balance between global search and local search
- Avoid the occurrence of premature convergence

Framework of our method (I)

The method of identifying nonlinear discrete-time systems can be divided into two parts:

1. The nonlinear discrete-time systems is approximated according to Volterra filter model with truncated second-order form. Moreover, the mean square error (MSE) is considered as the objective function to achieve the modeling requirement.

2. A ranked differential evolution algorithm is used to find suitable Volterra kernel vector ***H*** such that MSE is minimized.

Framework of our method (II)

1. Truncated second-order Volterra filter model

① The truncated second-order Volterra model is stated.
② The Volterra kernel vector in the truncated second-order Volterra model plays an important role in identifying nonlinear discrete-time systems.
③ To enable the model output to approximate the actual system output as much as possible, a suitable Volterra kernel vector should be determined.
④ The mean square error associated with the Volterra kernel vector is used for the designing requirement.

Framework of our method (III)

2. A ranked differential evolution algorithm (RDE)

① The RDE adjusts its scale factor dynamically.
② The RDE improves its mutation operation in terms of the ranked candidate solutions.
③ The RDE is used to optimize the mean square error (MSE).
④ When MSE is minimized by the RDE, an optimal approximation effect is obtained between the model output and the actual system output.

Major results

We compared six optimization algorithms on the identification problem of nonlinear discrete-time systems based on a Volterra filter model. To determine whether the results produced by the RDE algorithm were statistically different from those produced by the other five approaches, Wilcoxon rank-sum tests were conducted at the 5% significance level. The results are shown in Table 1. A *P*-value smaller than 0.05 suggests that the performance of the two approaches was statistically different with 95% certainty, whereas a *P*-value larger than 0.05 indicates no statistical difference.

Table 1 *P*-values from Wilcoxon rank-sum tests of performance results for six problems

Problem/Algorithm	PSO	IPSO	DE	IDE	SADE	RDE
Example 1a (N=5)	6.7956×10^{-8}	6.6909×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	NA
Example 1a (N=8)	6.7956×10^{-8}	6.7765×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	3.4156×10^{-7}	NA
Example 1b (N=5)	1.9177×10^{-7}	1.6571×10^{-7}	2.7451×10^{-4}	3.0566×10^{-3}	7.1135×10^{-3}	NA
Example 1b (N=8)	6.7956×10^{-8}	1.2009×10^{-6}	5.2269×10^{-7}	1.9883×10^{-1}	2.8530×10^{-1}	NA
Example 2a (N=8)	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	2.4706×10^{-4}	NA
Example 2b (N=8)	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	6.6737×10^{-6}	NA

"NA" stands for "not available"