HyperSoft Set Methods in Engineering

Journal Homepage: sciencesforce.com/hsse

HyperSoft Set Meth. Eng. Vol. 2 (2024) 104–113

Paper Type: Original Article

Neutrosophic Logic Applications in Intelligent Control Systems and its Stability

Nagarajan DeivanayagamPillai [1,*] and Said Broumi [2]

[1] Department of Mathematics, Rajalakshmi Institute of Technology, Chennai, Tamil Nadu, India; dnrmsu2002@yahoo.com.

[2] Laboratory of Information Processing, Faculty of Science Ben M'Sik, University Hassan II, Casablanca, Morocco; s.broumi@flbenmsik.ma.

Received: 10 Mar 2024 **Revised:** 30 Jun 2024 **Accepted:** 29 Jul 2024 **Published:** 01 Aug 2024

Abstract

Uncertainty is an inherent characteristic of actual control systems since their parameters are not fixed and may change based on external factors. The behaviour and performance of control systems can be strongly impacted by these unknown characteristics. Variability in the environment, ageing of core components, and manufacturing tolerances are some of the factors that contribute to this uncertainty and pose an unknown risk to control systems. As a result, real control systems' intrinsic parameter indeterminacy may have an impact on how well they operate. Real-life scenarios are often ambiguous and complex, posing challenges for decision-makers to articulate their opinions. Neutrosophic sets are employed in such contexts to handle indeterminacy effectively. Neutrosophic theory suggests a variable interval based on truth, indeterminacy, and falsity, allowing for the expression of both determinate and indeterminate information. Nevertheless, neutrosophic sets have not yet been used in the modelling, analysis, and design of uncertain control systems with determinate parameters. To close this gap, a new neutrosophic design technique is established in this study by introducing single-valued neutrosophic systems. The suggested control design method is illustrated with a numerical example employing neutrosophic parameters.

Keywords: Neutrosophic System, Neutrosophic Controller, Neutrosophic Stability.

1 | Introduction

Fuzzy Logic Control Systems (FLCSs) and Proportional Integrated Derivative (PID) controllers are the two main techniques used extensively in control systems. Despite being widely utilized in control applications, PID controllers struggle with nonlinear systems because they don't have enough parameter understanding. Researchers have concentrated on creating control systems that can manage unpredictable parameters to overcome this constraint. It has been determined that FLCSs are a good substitute for PID controllers because they provide faster settling times, less overshoot, improved stability, and fewer oscillations. Lotfi Zadeh developed fuzzy logic in 1965 as a means of managing uncertainties, and it has since grown to be a fundamental component of soft computing. Fuzzy theory offers a way to model linguistic terms like "many," "low," "medium," "often," and "few."

Uncertainty is a fundamental feature of real control systems since their parameters are not constants and can change based on external factors. As a result, the uncertain parameters of control systems might affect their behavior and performance. The plant's coefficients are usually regarded as determinate or nominal values in

Corresponding Author: dnrmsu2002@yahoo.com

https://doi.org/10.61356/j.hsse.2024.2348

traditional control scenarios. However, a variety of circumstances, including manufacturing tolerances, the aging of important components, and changes in the environment, can generate variances or uncertainties in the characteristics of the system, offering an undefined risk to the system. Analytical techniques and unique modeling are needed to guarantee the strong stability and control performance of systems with unknown parameters. Many modeling approaches for uncertain systems have been put forth; for example, testing the robust stability of dynamic electrical and mechanical systems can be done using interval linear time-invariant systems [1]. Fuzzy logic is extended by neutrophilic numbers (NN), which Smarandache devised to answer open problems [2-4]. NN adds an element of uncertainty to the model. This method works especially well when combining data from different sources of sensor data. Even though it is still in its infancy, neutrosophy has already garnered interest from research teams worldwide. Inference system enhancement models [5], [6]. It is currently one of the newest academic subjects with the fastest rate of global expansion, with applications in control theory, artificial intelligence, business, marketing, planning, and image processing. Because they greatly advance the state of the art in decision support systems and interact effectively with database expert systems, neuro philosophic reasoning components are especially well suited for robotic applications [7]. Neutrosophic logic is used in the Vladareanu-Smarandache approach for hybrid force-position robot control [8], which is well known for identifying angular errors in the actuator drive control loop of robot joints. A sort of logic known as neutrosophic logic assigns each proposition a percentage of falsity in subset F, a percentage of indeterminacy in subset I, and a percentage of truth in subset T [9]. The sets T, I, and F don't have to be intervals. The requirement that the membership and non-membership values in a given class add up to 1 [13-14], and [15] further restricts it. Comparable logics like Post's analysis of m values and Lukasiewicz logic's examination of three values (1, 1/2, and 0) are similarly crippled by the restriction that values can only move between 0 and 1[16]. To distinguish between relative and absolute truth, indeterminacy, or lie, there is no provision. It is impossible to definitively classify an element x on the borderline of a rough set as a member of a specific class or its complement, but can be very well described by neutrosophic logic, such that x (T, I, F) where T, I, F are standard or non-standard subsets of the non-standard interval]0- 1+[. In comparison to fuzzy counterparts, the proposed neutrosophic controller would be a unique system that is more generalized and indeterminacy-tolerant in its operation. The controllers for the proposed neutrosophic system would differ greatly based on the kind of control problems they are intended to solve. Here, we concentrate on elucidating relatively simple classifier problems. Neutrosophic systems can make use of information supplied by human operators, just like fuzzy systems do. Most real-world neutrosophic controllers are difficult to develop an accurate mathematical model to explain system behavior, and it is doubtful that the system will collect 100% comprehensive and conclusive data. Due to inherent nonlinearity, the time-varying nature of the process being managed, large unforeseen external disruptions, aging sensors, or other difficulties in generating precise measurements, the data may be incomplete or unclear. Under these conditions, human judgment can be quite helpful. Even though it is difficult to put control mechanisms into exact words, operators can frequently express them in vague language. Neutrosophic logic provides the important concept of neutrality range, whereas fuzzy logic controllers only consider if an element belongs to a particular class. The inherent ambiguity in data resulting from nonlinearity, outside disruptions, and measurement errors is ignored by this omission. As a result, fuzzy logic's applicability is restricted as it cannot completely account for membership and non-membership values. To tackle the issues of ambiguity and lacking information, a neutrosophic controller is suggested. Precise and partial information can be handled more effectively by neutrosophic logic, which goes beyond fuzzy logic in its typical applications. While many researchers have confirmed that fuzzy logic can provide. Compared to its fuzzy equivalents, the proposed neutrosophic controller represents a specialized system that is more generalized and tolerant of indeterminacy. A neutrosophic set with values for falsehood, indeterminacy, and truth is represented by the Neutrosophic Set Class.

Membership Functions: Basic functions to determine the truth, indeterminacy, and falsity of membership values.

Convert a measurement into a Neutrosophic Set by creating a The controllers for the proposed neutrosophic systems will differ greatly based on the particular control problems they are intended to solve. We address several relatively simple categorization problems in this topic. Like its fuzzy siblings, neutrosophic systems are likewise capable of making good use of human skill. It is difficult to create a mathematical model that accurately represents the majority of controllers in the actual world because it is improbable that complete and accurate data will be obtained. The existence of intrinsic non-linearity, the regulated process's time-varying nature, significant and unforeseen outside disruptions, failing sensors, and other problems. It can be useful to use human judgment in some circumstances. Operators can typically describe the control mechanism in relatively simple words, even when it is challenging to convey this knowledge precisely.

2 | Algorithms

Keep a record of all measurements taken for each variable related to the regulated process.

The measurement's truth, falsity, and indeterminacy. This conversion process is known as neutrosophication.

The inference engine then evaluates the control rules stored in the neutrosophic rule base using neutrosophic metrics. This evaluation generates one or more neutrosophic sets defined over the universe of possible actions.

In the final phase of the process, the resulting neutrosophic set is converted into a single (crisp) value in a triplet format, such as x(t,I,f)which provides the most precise representation. This conversion process is known as de-neutrosophication in Figure 1.

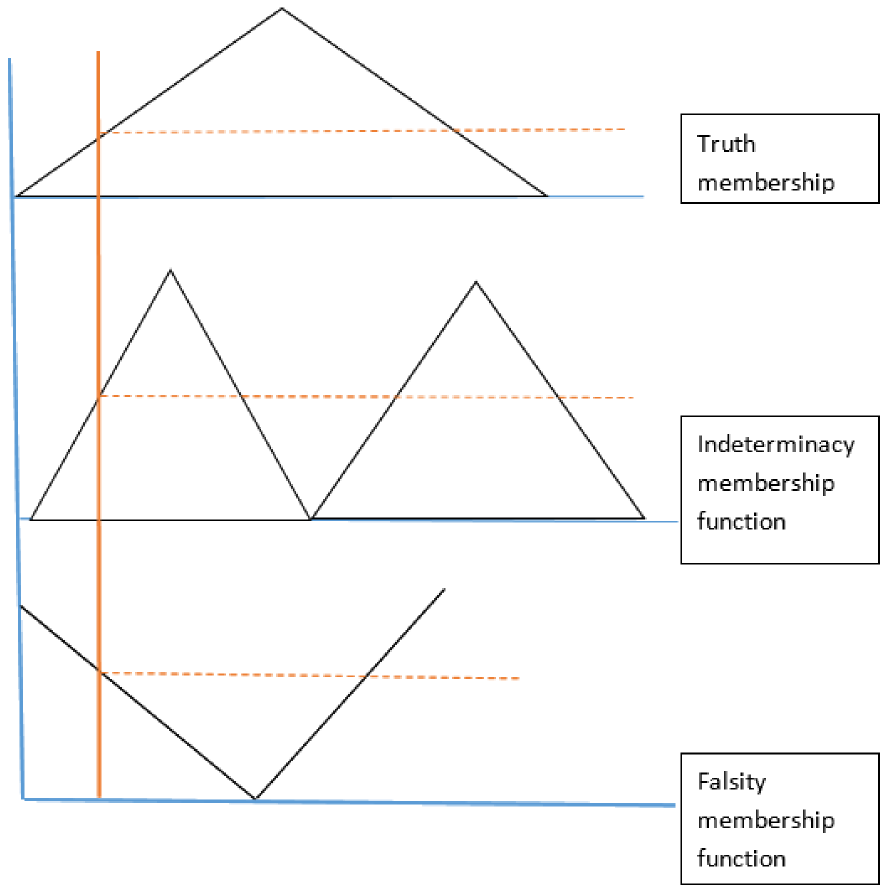

Truth membership
Indeterminacy membership function
Falsity membership function

Figure 1. Neutrosophication process.

There are several important steps involved in creating a Python neutrosophic control system algorithm. The algorithm must produce control actions, manage neutrosophic sets, and carry out inference using neutrosophic logic. An abridged Python algorithm for a fundamental neutrosophic control system can be seen below. This example presumes a rudimentary knowledge of fuzzy logic, control systems, and neutrosophic logic.

A neutrosophic set with values for falsehood, indeterminacy, and truth is represented by the neutrosophic set class.

- Membership Functions: Basic functions to determine the truth, indeterminacy, and falsity of membership values.
- Neutrosophic Set Creation: Using membership functions, this function transforms a measurement into a neutrosophic set.
- Evaluate Rules: Creates an output neutrosophic set by applying a set of rules to the input neutrosophic set.
- De-neutrosophic: Uses the truth component in this instance to reduce the neutrosophic set to a single, crisp value.
- Neutrosophic Control System: Combines the aforementioned features to imitate a fundamental control scheme using neutrosophic reasoning.

Neutrosophic Control System Algorithm in Python program in Appendix 1.

Output is Control action based on measurement 0.7: 0.2.

3 | Neutrosophic Control System Stability Algorithm in Python

This code defines a neutrosophic set class and uses it to simulate a control system's stability analysis. Here's a breakdown:

- NeutrosophicSet class: Represents a neutrosophic set with truth, indeterminacy, and falsity values.
- Membership functions: Defines truth, indeterminacy, and falsity membership functions for creating neutrosophic sets.
- Create_neutrosophic_set: Creates a neutrosophic set from a measurement value.
- Evaluate_stability: Evaluates stability based on the neutrosophic set (placeholder logic, assumes stability relates to truth and indeterminacy).
- Check_stability: Checks if the stability index exceeds a threshold.
- Neutrosophic_control_system_stability: Simulates the control system, creating neutrosophic sets, evaluating stability, and checking stability criteria for each measurement.
- Example usage: Generates 50 measurements, runs the simulation, and plots the stability index against measurements, with a stability threshold line.

The plot shows how the stability index changes with measurements, indicating stable (above threshold) and unstable regions. This demonstrates a basic neutrosophic control system stability analysis.

Please note that this is a simplified example. In real-world applications, the stability evaluation logic and membership functions would be more complex and domain-specific in Appendix 2

The output of the code is a plot showing the stability index of a control system against various measurements. The plot has the following features:

- X-axis: Measurements (ranging from 0 to 1).
- Y-axis: Stability Index (ranging from 0 to 1).

- Blue line: Stability index values for each measurement.
- Red dashed line: Stability Threshold (set at 0.1).

The plot indicates the stability of the control system for each measurement. Measurements with a stability index above the threshold (0.1) are considered stable, while those below the threshold are considered unstable and the stability results in Table 1.

Here's a sample output:

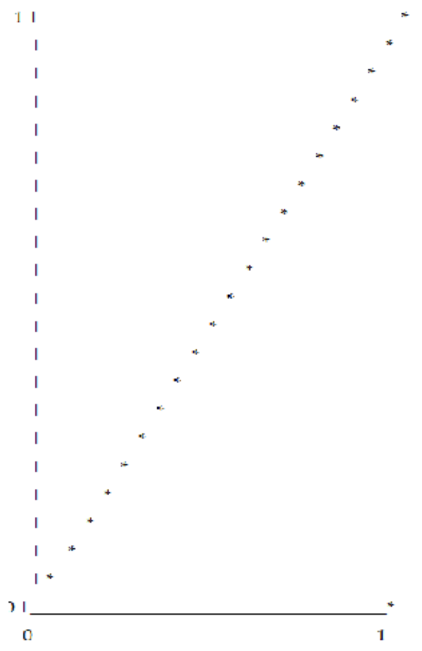
Table 1. Stability results.

Measurement	Stability Index	Is Stable
0.0	0.9	True
0.1	0.8	True
0.2	0.7	True
...	...	...
0.8	0.2	False
0.9	0.1	False
1.0	0.0	False

The plot and stability results demonstrate how the neutrosophic control system stability analysis works, showing the transition from stable to unstable regions as the measurements change.

Graph Description: The graph shows the stability index of a control system against various measurements. The x-axis represents the measurements ranging from 0 to 1, and the y-axis represents the stability index ranging from 0 to 1. A blue line plots the stability index values for each measurement, and a red dashed line indicates the stability threshold set at 0.1.

ASCII Art Representation:



In this ASCII art representation, the * symbols form a line representing the stability index values, and the * symbol on the y-axis indicates the stability threshold.

The code demonstrates a basic implementation of a neutrosophic control system stability analysis using Python. The neutrosophic set theory is used to handle uncertainty and indeterminacy in the system, and the stability index is calculated based on the truth, indeterminacy, and falsity values.

The graph shows the stability index of the control system against various measurements, indicating stable and unstable regions. The stability threshold is set at 0.1, and measurements with a stability index above this threshold are considered stable.

4 | Conclusion

While intuitionistic fuzzy sets (IFS) and interval-valued intuitionistic fuzzy sets (IVIFS) have been effective in capturing the falsity of beliefs, their applications are limited by the constraint of values between 0 and 1. This limitation shows that while IFS and IVIFS are well-suited to address incompleteness, they struggle with inconsistent and indeterminate information commonly encountered in belief systems. To address this, we propose fuzzy models that leverage the strengths of neutrosophic logic. Our research has developed simple neutrosophic classifiers for data classification. Neutrosophic logic, with its inherent generality and flexibility in extending beyond the [0, 1] range, is poised to benefit various fields where information is vague, uncertain, partial, or contradictory.

Acknowledgments

The author is grateful to the editorial and reviewers, as well as the correspondent author, who offered assistance in the form of advice, assessment, and checking during the study period.

Author Contributions

All authors contributed equally to this work.

Funding

This research did not receive any external funding.

Conflicts of Interest

The authors declare that there is no conflict of interest in the research.

Ethical Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Data Availability

There was no data used in the inquiry that was as stated in the article.

References

[1] Hussein, M. T. (2015). Modeling mechanical and electrical uncertain systems using functions of robust control MATLAB Toolbox® 3. International Journal of Advanced Computer Science and Applications, 6(4), 79-84.

[2] Smarandache, F. (2005). A unifying field in logics: neutrosophic logic. Neutrosophy, neutrosophic set, neutrosophic probability: neutrsophic logic. Neutrosophy, neutrosophic set, neutrosophic probability. Infinite Study.

[3] Smarandache, F. (2013). Introduction to neutrosophic measure, neutrosophic integral, and neutrosophic probability. Infinite Study.

[4] Smarandache, F. (2014). Introduction to Neutrosophic Statistics, Sitech & Education Publishing, Craiova, 2014, 124 p.

[5] Smarandache, F. (1999). A unifying field in Logics: Neutrosophic Logic. In Philosophy (pp. 1-141). American Research Press.

[6] Vladareanu, V., Munteanu, R. I., Mumtaz, A., Smarandache, F., & Vladareanu, L. (2015). The optimization of intelligent control interfaces using Versatile Intelligent Portable Robot Platform. Procedia Computer Science, 65, 225-232.

[7] Smarandache, F. (2014). Neutrosophic Theory and Its Applications, Vol. I: Collected Papers. Infinite Study.

[8] Smarandache, F., & Vlădăreanu, L. (2011, November). Applications of neutrosophic logic to robotics: An introduction. In 2011 IEEE international conference on granular computing (pp. 607-612). IEEE.
[9] Smarandache, F. (2013, September). Definiton of neutrosophic logic-a generalization of the intuitionistic fuzzy logic. In Conference proceedings, pages (Vol. 141, p. 146). in: Conference proceedings, pages 141-146.
[10] Smarandache, F. (1999). A unifying field in Logics: Neutrosophic Logic. In Philosophy (pp. 1-141). American Research Press.
[11] Smarandach, F. (2002). A new branch of philosophy, in multiple valued logic. An international journal, 8(3), 297-384.
[12] Smarandache, F. (2003). Proceedings of the First International Conference on Neutrosophy, Neutrosophic Logic, Neutrosophic Set, Neutrosophic Porbability and Statistics: Www. Gallup. Unm. Edu/~ Smarandache/NeutrosophicProceedings. Pdf. Infinite Study.
[13] Broumi, S., Nagarajan, D., Lathamaheswari, M., Talea, M., Bakali, A., & Smarandache, F. (2020). Intelligent algorithm for trapezoidal interval valued neutrosophic network analysis. CAAI Transactions on Intelligence Technology, 5(2), 88-93.
[14] Lathamaheswari, M., Nagarajan, D., Kavikumar, J., & Phang, C. (2018). A review on type-2 fuzzy controller on control system. Journal of Advanced Research in Dynamical and Control Systems, 10(11), 430-435.
[15] Amador-Angulo, L., Castillo, O., Castro, J. R., & Melin, P. (2023). A new approach for interval type-3 fuzzy control of nonlinear plants. International Journal of Fuzzy Systems, 25(4), 1624-1642.
[16] Khan, M. J., Alcantud, J. C. R., Kumam, W., Kumam, P., & Alreshidi, N. A. (2023). Expanding Pythagorean fuzzy sets with distinctive radii: Disc Pythagorean fuzzy sets. Complex & Intelligent Systems, 9(6), 7037-7054.

Appendix 1

```
import numpy as np

# Define a Neutrosophic Set class

class NeutrosophicSet:

    def __init__(self, truth, indeterminacy, falsity):

        self.truth = truth

        self.indeterminacy = indeterminacy

        self.falsity = falsity

    def __repr__(self):

        return f"NeutrosophicSet(truth={self.truth}, indeterminacy={self.indeterminacy}, falsity={self.falsity})"

# Example membership functions for truth, indeterminacy, and falsity

def truth_membership(value):

    return max(0, min(1, 1 - value))

def indeterminacy_membership(value):

    return max(0, min(1, value))

def falsity_membership(value):

    return max(0, min(1, 1 - value))

# Define a function to create a neutrosophic set from measurements

def create_neutrosophic_set(measurement, scale=1.0):

    truth = truth_membership(measurement / scale)

    indeterminacy = indeterminacy_membership(measurement / scale)

    falsity = falsity_membership(measurement / scale)
    return NeutrosophicSet(truth, indeterminacy, falsity)
# Example rule base (simple rules for demonstration)
def evaluate_rules(input_set):
    # Example rules, modify based on your actual rules
    if input_set.truth > 0.5:
        return NeutrosophicSet(truth=0.8, indeterminacy=0.1, falsity=0.1)
    else:
        return NeutrosophicSet(truth=0.2, indeterminacy=0.6, falsity=0.2)
# Defuzzification function
def de_neutrosophicate(neutrosophic_set):
    # Example defuzzification, return the truth component
    return neutrosophic_set.truth
# Main control system function
def neutrosophic_control_system(measurement):
    # Create neutrosophic set from measurement
    input_set = create_neutrosophic_set(measurement)
    # Evaluate rules to generate control action
    output_set = evaluate_rules(input_set)
    # Defuzzify to get a single control value
    control_value = de_neutrosophicate(output_set)
    return control_value
# Example usage
measurement = 0.7
control_action = neutrosophic_control_system(measurement)
print(f"Control action based on measurement {measurement}: {control_action}")
```

Appendix 2

```
import numpy as np
import matplotlib.pyplot as plt
# Define the NeutrosophicSet class
class NeutrosophicSet:
    def _init_(self, truth, indeterminacy, falsity):
        self.truth = truth
        self.indeterminacy = indeterminacy
        self.falsity = falsity
    def _repr_(self):
        return f"NeutrosophicSet(truth={self.truth}, indeterminacy={self.indeterminacy},
falsity={self.falsity})"
# Define membership functions
def truth_membership(value):
    return max(0, min(1, 1 - value))
def indeterminacy_membership(value):
    return max(0, min(1, value))
def falsity_membership(value):
    return max(0, min(1, 1 - value))
# Function to create a neutrosophic set from measurements
def create_neutrosophic_set(measurement, scale=1.0):
    truth = truth_membership(measurement / scale)
    indeterminacy = indeterminacy_membership(measurement / scale)
    falsity = falsity_membership(measurement / scale)
    return NeutrosophicSet(truth, indeterminacy, falsity)
# Evaluate stability based on neutrosophic sets
def evaluate_stability(input_set):
    # Placeholder for stability evaluation logic
    # For demonstration, assume stability is related to the truth and indeterminacy values
    stability_index = input_set.truth - input_set.indeterminacy
    return stability_index
# Check stability criteria
def check_stability(stability_index, threshold=0.1):
    return stability_index > threshold
# Main function to simulate the control system and stability
def neutrosophic_control_system_stability(measurements):
    stability_results = []
    for measurement in measurements:
        input_set = create_neutrosophic_set(measurement)
        stability_index = evaluate_stability(input_set)
        is_stable = check_stability(stability_index)
        stability_results.append((measurement, stability_index, is_stable))
    return stability_results
# Example usage
measurements = np.linspace(0, 1, 50)  # Create 50 measurements from 0 to 1
results = neutrosophic_control_system_stability(measurements)
# Extracting data for plotting
measurements, indices, stabilities = zip(*results)
# Plotting the results
plt.figure(figsize=(10, 6))
plt.plot(measurements, indices, 'o-', label='Stability Index', color='b')
plt.axhline(y=0.1, color='r', linestyle='--', label='Stability Threshold')
plt.xlabel('Measurement')
plt.ylabel('Stability Index')
plt.title('Neutrosophic Control System Stability Analysis')
plt.legend()
plt.grid(True)
plt.show()
```