UTM UNIVERSITI TEKNOLOGI MALAYSIA | Faculty of Artificial Intelligence

FUZZY LOGIC CONTROLLER APPLICATION:

LOAN APPROVAL SYSTEM

SAIA 1193

COMPUTATIONAL INTELLIGENCE

SEMESTER 2024/2025-2

SUBMISSION DATE: 27 JUNE 2025

GROUP MEMBERS	1. **AMINA ASYIFFA BINTI ASPIYAH MAHYUS (A24AI0015)** 2. **FARIN BATRISYIA BINTI SAIPUL NIZAM (A24AI0030)** 3. **LE YONG XIANG (A24AI0045)** 4. **MUHAMMAD DANISH IQBAL BIN MOHAMAD HASSAN (A24AI0052)** 5. **WAN ALIF DANIAL BIN WAN KAMARULFARID (A24AI0093)**

1. Introduction

The loan approval process in financial institutions involves complex decision-making that considers multiple factors, including creditworthiness, financial stability, and risk assessment. However, according to Angraini, Rosalina, and Kosasih (2024), credit score and debt-to-income ratio are the most influential factors. Traditional binary decision systems often fail to capture the nuanced nature of loan evaluation, where factors exist on a spectrum rather than in discrete categories.

This project implements a **Fuzzy Logic Controller (FLC)** for automated loan approval decisions. Fuzzy logic is particularly suitable for this application because:

1. **Uncertainty Handling**: Loan decisions involve uncertainty and imprecise information
2. **Human-like Reasoning**: Mimics how loan officers make decisions using linguistic terms
3. **Gradual Transitions**: Allows smooth transitions between approval categories
4. **Multiple Criteria**: Effectively handles multiple conflicting criteria

Our system evaluates loan applications based on four key input variables and provides two output decisions: approval status and recommended interest rate.

2. System Design

The primary goal of this project is to develop a robust Fuzzy Logic Controller (FLC) specifically tailored for a complex loan approval system. This architecture effectively simulates human expert decision-making by addressing the inherent uncertainties and complexities in financial data. The system will accept multiple input criteria related to an applicant's financial profile and deliver precise, continuous outputs regarding loan approval status and recommended interest rates.

2.1. Input Variables

To fully assess loan applications, the FLC includes four important input variables, each with a specified universe of terms based on common banking standards and financial risk assessment practices:

Input Variables	Linguistic Terms	Min – Max values of Variables
Credit Score	Poor / Fair / Good / Excellent	300 - 850
Debt Ratio	Low / Medium / High	0 - 100
Annual Income	Low / Medium / High	0 - 200000
Employment Duration	Short / Medium / Long	0 - 40

The FLC utilizes four critical input variables that comprehensively capture an applicant's financial profile and creditworthiness. The Credit Score serves as a key indicator of an applicant's creditworthiness, reflecting their historical ability to manage debt obligations responsibly, which lenders assess when reviewing loan applications (Sree, Srilatha, & Prasuna, 2023). This variable follows the standardized FICO scoring model, ranging from 300 to 850, where higher scores indicate better credit management and lower default risk. The Debt-to-Income Ratio (DTI) provides insight into an applicant's current financial burden by measuring the percentage of their gross monthly income allocated to debt payments. With a theoretical range of 0% to 100%, this metric indicates whether an applicant has sufficient disposable income to take on additional debt obligations, as lower ratios directly correlate with improved financial health and repayment capacity (Durojaiye, O., & Sahi, 2024).

The Annual Income variable quantifies the applicant's total yearly earnings, establishing their fundamental capacity to service loan payments. The system accommodates a wide economic spectrum ranging from $0 to $200,000, including various socioeconomic segments from entry-level workers to high-income professionals. This comprehensive range ensures the system can evaluate applications across diverse income groups while maintaining appropriate risk assessment standards. Finally, Employment Duration measures the length of continuous employment, spanning from 0 to 40 years, and is a vital indicator of stability. This variable reflects job security and income predictability, with longer employment periods typically correlating with reduced default risk due to established career stability and consistent income streams (Kuhn & Ploj, 2020). Together, these four variables create a holistic financial profile that enables the fuzzy logic system to make nuanced loan approval decisions based on multiple complementary risk factors.

2.2. Output Variables

The FLC generates two primary outputs that collectively determine the outcome of a loan application:

Output Variables	Linguistic Terms	Min – Max values of Variables
Loan Approval Score	Reject / Review / Approve	0 - 100
Interest Rate	Low / Medium / High	3% - 25%

The FLC generates two interconnected output variables that deliver comprehensive guidance for loan decisions. The Approval Score provides a continuous numerical assessment of loan approval likelihood, utilizing a standardized scale from 0 to 100, where the extremes indicate definitive decisions and intermediate values reflect specific levels of approval confidence. A score of 0 represents outright rejection, typically reserved for applications with critical risk factors such as poor credit history or excessive debt burdens, while a score of 100 signifies full approval with minimal perceived risk. The intermediate range allows for nuanced decision-making, where scores can be systematically mapped to actionable linguistic categories: low scores (0-35) correspond to 'Rejected' applications, middle-range scores (20-80) indicate 'Review' cases requiring human oversight, and high scores (65-100) translate to 'Approved' applications.

The **Recommended Interest Rate** output implements risk-based pricing directly tied to the approval decision, offering a range of 3% to 25% annual percentage rate to precisely match the full spectrum of applicant risk profiles. The lower bound of 3% represents prime lending rates reserved for exceptional candidates with excellent credit scores, stable employment, and minimal debt obligations, reflecting the institution's confidence in repayment reliability. Conversely, the upper bound of 25% addresses high-risk scenarios where applicants may have suboptimal credit histories or elevated debt-to-income ratios but still qualify for lending under appropriate risk compensation.

2.3. System Architecture

The FLC's design follows a standard four-component architecture: Fuzzification, Rule Base, Inference Engine, and Defuzzification. A high-level block diagram illustrating this architecture for the Loan Approval System is shown below.

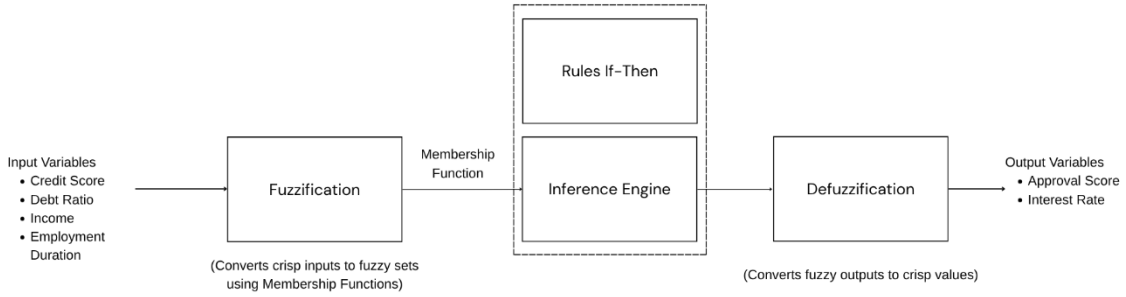

Figure 1: High-level architecture of the fuzzy logic loan approval system

2.3.1. Fuzzification Stage

The fuzzification component serves as the system's input interface, converting crisp numerical values into fuzzy membership degrees (Sope & Fujio, 2023). The four input variables—Credit Score, Debt Ratio, Income, and Employment Duration—enter this stage as precise measurements and are transformed into linguistic fuzzy sets using predefined membership functions. For instance, a credit score of 720 might simultaneously belong to both "Good" (0.7 membership) and "Excellent" (0.3 membership) categories, capturing the nuanced nature of creditworthiness assessment. This parallel membership allows the system to handle borderline cases more effectively than traditional binary classification systems.

2.3.2. Rule Base Component

The Rule Base houses the domain expertise in the form of structured IF-THEN statements, representing the core knowledge repository of the system. These eight carefully constructed rules encode typical banking practices and risk assessment principles that establish logical relationships between input combinations and desired outputs. The rules operate on linguistic terms rather than numerical values, making them intuitive and easily interpretable. This knowledge representation approach facilitates system maintenance and rule modification, as banking policies can be updated by adjusting linguistic rules.

2.3.3. Inference Engine

The inference engine is a critical component of a fuzzy logic system, as it processes and interprets complex data by applying fuzzy logic principles to transform input into output, directly influencing the whole performance of the system (Kgatwe et al., 2023). This component employs Mamdani inference methodology, calculating rule activation strengths using minimum operations for AND conditions and maximum operations for OR conditions. The engine evaluates all applicable rules simultaneously, determining the degree to which each rule fires based on the input membership values. Subsequently, it aggregates the rule outputs using maximum composition to ensure the strongest applicable rule influences each output category. This parallel processing approach captures the complex interactions between multiple risk factors while maintaining computational efficiency.

2.3.4. Defuzzification Stage

The Defuzzification component completes the transformation cycle by converting the aggregated fuzzy output sets back into crisp numerical values suitable for practical application. Using the centroid method, this stage calculates the center of gravity of the combined output membership functions to produce definitive Approval Scores (0-100) and Recommended Interest Rates (3-25%). The centroid method, a popular defuzzification technique, computes a weighted average from the domain of the membership function to produce a smooth and continuous output that represents the relative importance of all activated rules. This solution efficiently addresses the abrupt transitions inherent in traditional rule-based systems, delivering structured and consistent output (Mitsuishi, 2022).

2.3.5. System Integration and Data Flow

The architectural design ensures seamless data flow between components while maintaining modularity and extensibility. Input variables flow unidirectionally through the system, preventing feedback loops that could introduce instability. The membership function definitions act as a crucial interface between fuzzification and defuzzification stages, ensuring consistent interpretation of linguistic terms throughout the process. This modular approach enables the modification or enhancement of individual components without compromising overall system functionality, facilitating long-term maintenance and system evolution as lending practices and regulatory requirements change.

3. Fuzzy Rule Base

The fuzzy rule base serves as the FLC's operational core, encapsulating expert knowledge for loan approval decisions in a series of "IF-THEN" statements. These rules convert different combinations of fuzzy input linguistic terms (e.g., 'good credit score', 'low debt ratio') to fuzzy output linguistic terms (e.g., 'high approval', 'low interest rate'). The development of these rules is crucial since they directly influence the system's decision-making behavior.

3.1. Rule Derivation and Rationale

The eight rules in our system (described in Appendix A) were developed through a simulation of typical banking practices and risk assessment concepts. The aim was to develop a rule system that reflected how a human loan officer would evaluate various application details.

- **Prioritizing Risk Factors**: Rules prioritize critical financial health indicators. For example, a “Poor” Credit Score (Rule 6) or a “High” Debt Ratio (Rule 7) immediately leads to a “Reject” decision and “High” interest rate, regardless of other positive factors. This demonstrates the high-risk character of these indicators
- **Income and Employment Stability**: Low income (Rule 8) and short employment duration are significant red flags, often leading to rejection, as they imply financial instability.
- **Balancing Positive Attributes**: For applicants with “Good” or “Excellent” credit scores (Rules 1, 2, 3), the decision strongly favors “Approve” with “Low” and “Medium” interest rates, assuming that other factors, such as debt ratio, are also favorable.
- **“Fair” Category and “Review” Decision**: The “Fair” credit score category (Rules 4, 5) often results in a “Review” decision. This acknowledges that although not ideal, such applicants can still be approved if other factors are strong, such as low debt and long employment. This mirrors real-world scenarios where borderline cases require further manual review. The “Review” output in our system translates to an approval score that falls in the mid-range, indicating uncertainty that might warrant human intervention.
- **Handling “Any” Conditions**: The use of “Any” for certain input variables in some rules (e.g. Rule: IF Credit Score is Excellent AND Debt Ratio is Low THEN Approval is Approve AND Interest Rate is Low) signifies that these particular inputs are dominant and override the influence of other variables for that specific rule, simplifying the rule base while capturing strong correlations.

This structured approach ensures that the fuzzy logic system makes logical and consistent loan approval decisions, aligning with prudent financial management principles.

4. Implementation

The FLC for the Loan Approval System is implemented in Python, utilizing the scikit-fuzzy library for optimized fuzzy logic computations. The implementation follows a modular, object-oriented approach, encapsulating the entire FLC within a FuzzyLoanController class. This design promotes maintainability, reusability and scalability.

4.1. Class Initialization and Setup (FuzzyLoanController)

The FuzzyLoanController class constructor (__init__) is responsible for defining the universe of discourse (ranges) for all input and output variables and setting up the membership functions and the control system.

```
class FuzzyLoanController:
    def __init__(self):
        # Define input variable ranges
        self.credit_score_range = (300, 850)
        self.debt_ratio_range = (0, 100)  # percentage
        self.income_range = (0, 200000)  # annual income in dollars
        self.employment_duration_range = (0, 40)  # years

        # Define output variable ranges
        self.approval_score_range = (0, 100)  # 0=reject, 100=approve
        self.interest_rate_range = (3, 25)  # percentage
```

Figure 2: FuzzyLoanController class

The FuzzyLoanController class contains a constructor to initialize the fuzzy controller with predefined universes of discourse for all variables. Then, it establishes all of the input variables' boundaries based on banking industry standards, where the credit scores are in the range of 300 to 850, the debt ratio is a percentage from 0 to 100%, the income is up to $200,000, and the employment duration is up to 40 years. It also defines clear decision boundaries for output ranges, including approval scores (0-100) and interest rate (3-25%). To make it easy for parameter modification and system scalability, it has been designed with the use of instance variables.

4.2. Membership Functions Definition

Membership functions define how crisp input values are mapped to fuzzy linguistic terms. Both triangular and trapezoidal membership functions are utilized to represent the varying degrees of membership for each linguistic term (e.g., “Low”, “Medium”, “Poor”, “Fair”, “Good”, “Excellent”). These functions are an important component before proceeding to the fuzzification process.

4.2.1. Credit Score Membership Functions

```python
def get_credit_score_membership(self, score: float) -> Dict[str, float]:
    """Define membership functions for credit score"""
    return {
        'poor': self.trapezoidal_membership(score, 300, 300, 500, 580),
        'fair': self.triangular_membership(score, 500, 620, 720),
        'good': self.triangular_membership(score, 650, 720, 780),
        'excellent': self.trapezoidal_membership(score, 720, 800, 850, 850)
    }
```

Figure 3: get_credit_score_membership method

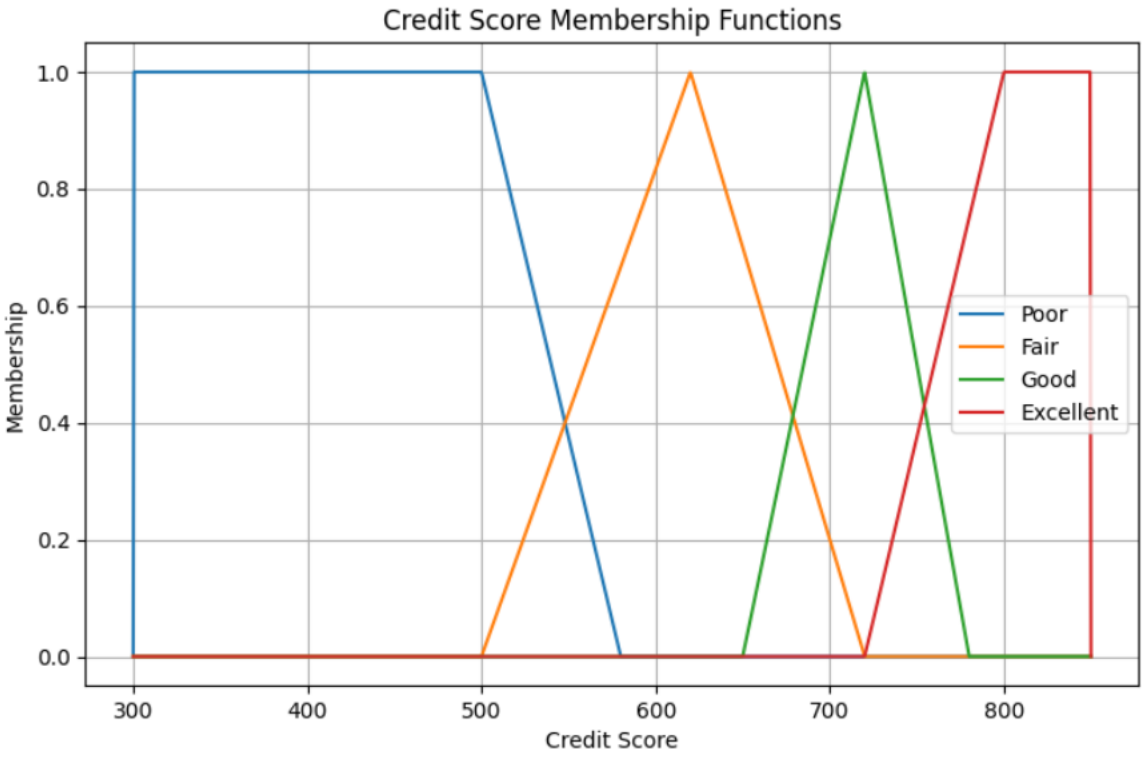

Figure 4: Membership functions for credit score

- **Poor (300-580)**: Trapezoidal with flat bottom (300-500) ensures all scores ≤300 are fully "poor"

- **Fair (500-720)**: Triangular peaks at 620, overlaps with poor (500-580) and good (650-720)
- **Good (650-780)**: Triangular peaks at 720, creates smooth transition zones
- **Excellent (720-850)**: Trapezoidal with flat top (800-850) for premium scores
- **Overlap Strategy**: Intentional overlaps prevent sharp decision boundaries

4.2.2. Debt-to-Income Ratio Membership Functions

```python
def get_debt_ratio_membership(self, ratio: float) -> Dict[str, float]:
    """Define membership functions for debt-to-income ratio"""
    return {
        'low': self.trapezoidal_membership(ratio, 0, 0, 20, 35),
        'medium': self.triangular_membership(ratio, 25, 40, 55),
        'high': self.trapezoidal_membership(ratio, 45, 60, 100, 100)
    }
```

Figure 5: get_debt_ratio_membership method

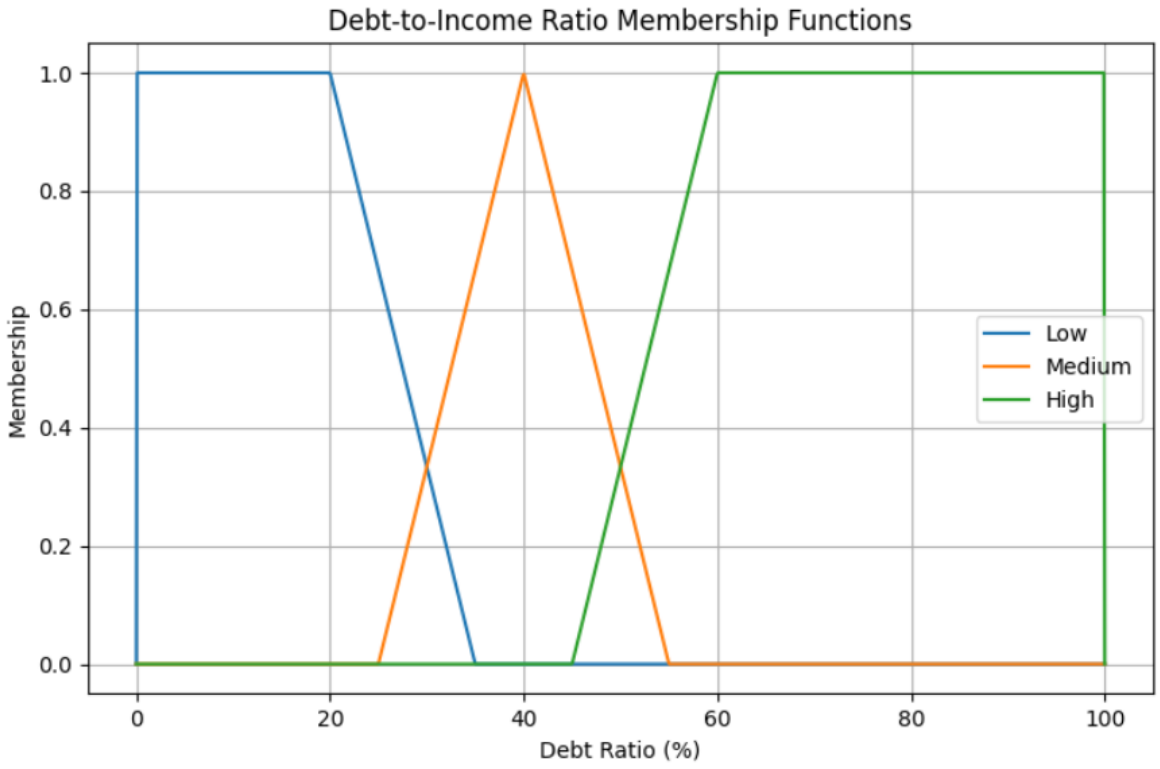

Figure 6: Debt-to-Income Ratio Membership Functions

- **Low (0-35%)**: Conservative lending standard, flat region 0-20% for excellent ratios
- **Medium (25-55%)**: Evaluation zone around 40% (typical lending threshold)
- **High (45-100%)**: Risk zone, flat top 60-100% ensures all high ratios are flagged

- **Critical Threshold**: 43% DTI is the industry standard maximum, captured in medium-high overlap

4.2.3. Annual Income Membership Functions

```python
def get_income_membership(self, income: float) -> Dict[str, float]:
    """Define membership functions for annual income"""
    return {
        'low': self.trapezoidal_membership(income, 0, 0, 30000, 50000),
        'medium': self.triangular_membership(income, 35000, 70000, 120000),
        'high': self.trapezoidal_membership(income, 80000, 150000, 200000, 200000)
    }
```

Figure 7: get_income_membership method

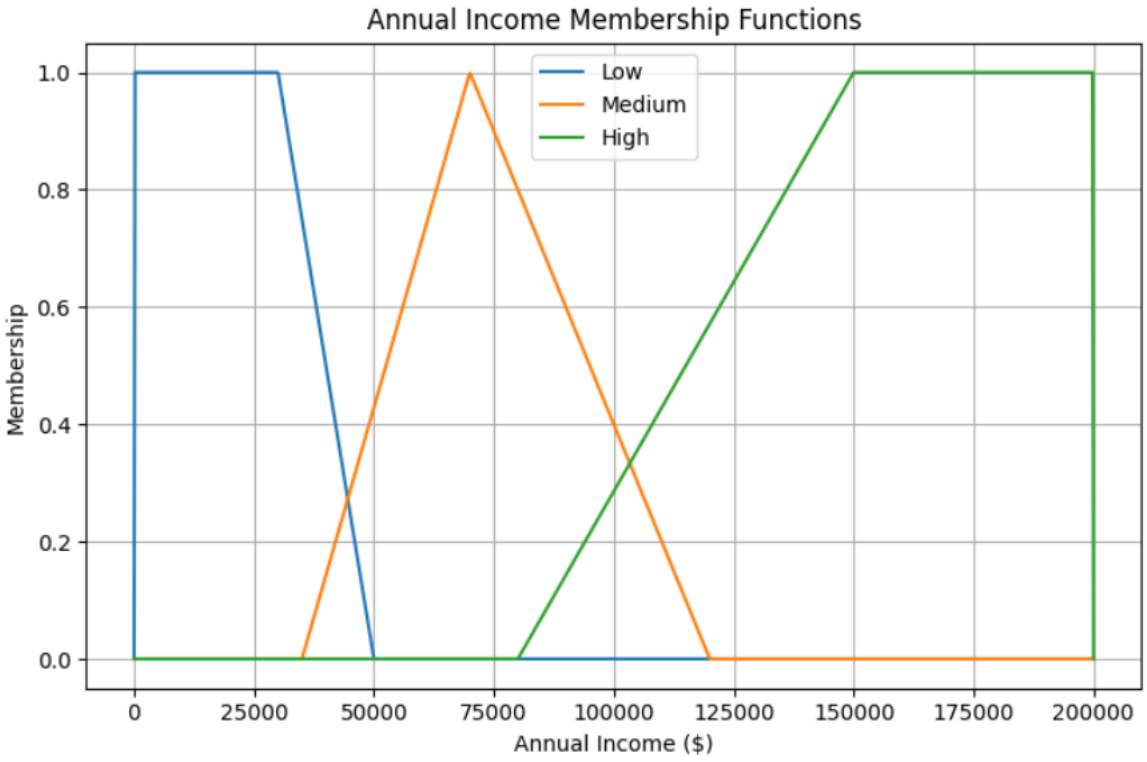

Figure 8: Annual Income Membership Functions

- **Low (≤$50k)**: Below median household income, flat region 0-30k for financial hardship cases
- **Medium ($35k-$120k)**: Middle class bracket, peaks at $70k (near US median)
- **High ($80k+)**: Above-average earners, flat top $150k+ for high earners
- **Overlap Design**: Smooth transitions accommodate regional income variations

4.2.4. Employment Duration Membership Functions

```python
def get_employment_membership(self, duration: float) -> Dict[str, float]:
    """Define membership functions for employment duration"""
    return {
        'short': self.trapezoidal_membership(duration, 0, 0, 1, 3),
        'medium': self.triangular_membership(duration, 2, 5, 10),
        'long': self.trapezoidal_membership(duration, 7, 15, 40, 40)
    }
```

Figure 9: get_employment_membership method

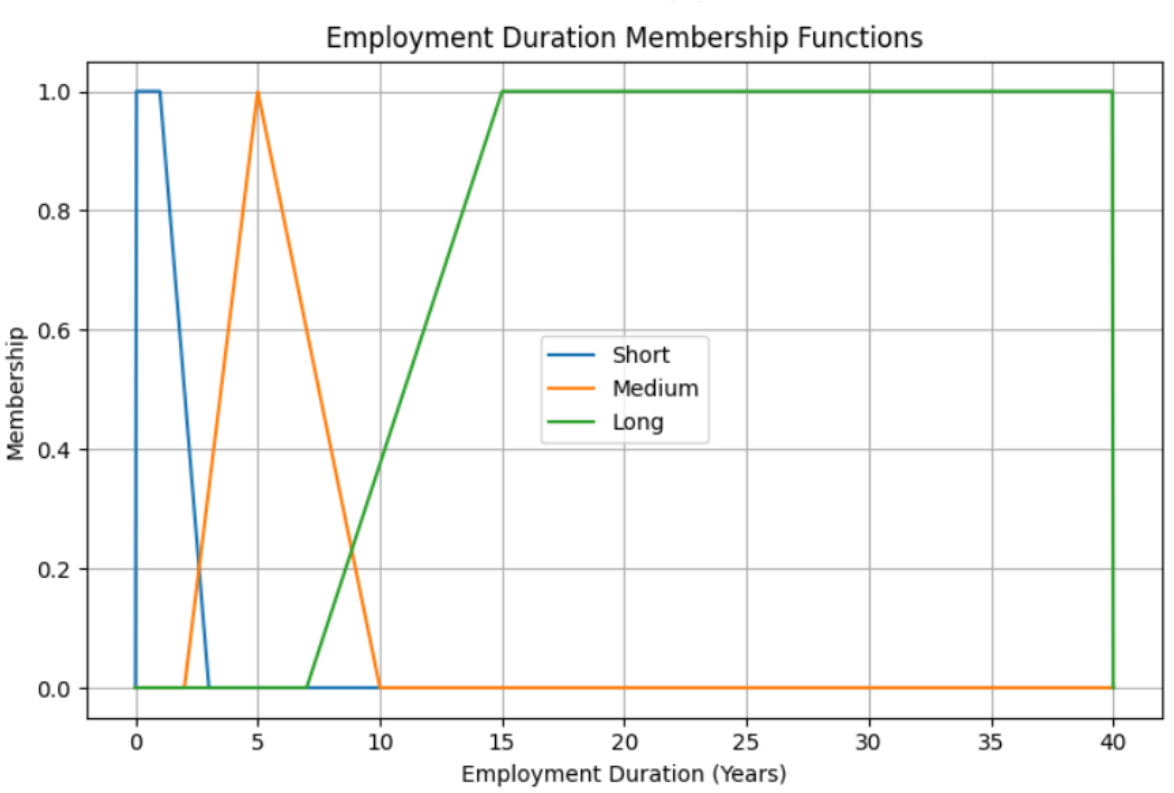

Figure 10: Employment Duration Membership Function

- **Short (0-3 years)**: Job instability period, flat region 0-1 year for new employees
- **Medium (2-10 years)**: Established employment, peaks at 5 years (career stability indicator)
- **Long (7+ years)**: High stability, flat top 15+ years for career veterans
- **Risk Assessment**: Longer employment suggests lower default risk

4.2.5. Approval Score Membership Functions

```python
def get_approval_membership_inverse(self, approval_level: str) -> Tuple[float, float, float, float]:
    """Get parameters for output membership functions"""
    if approval_level == 'reject':
        return (0, 0, 15, 35)
    elif approval_level == 'review':
        return (20, 40, 60, 80)
    elif approval_level == 'approve':
        return (65, 85, 100, 100)
```

Figure 11: get_approval_memberships_inverse method

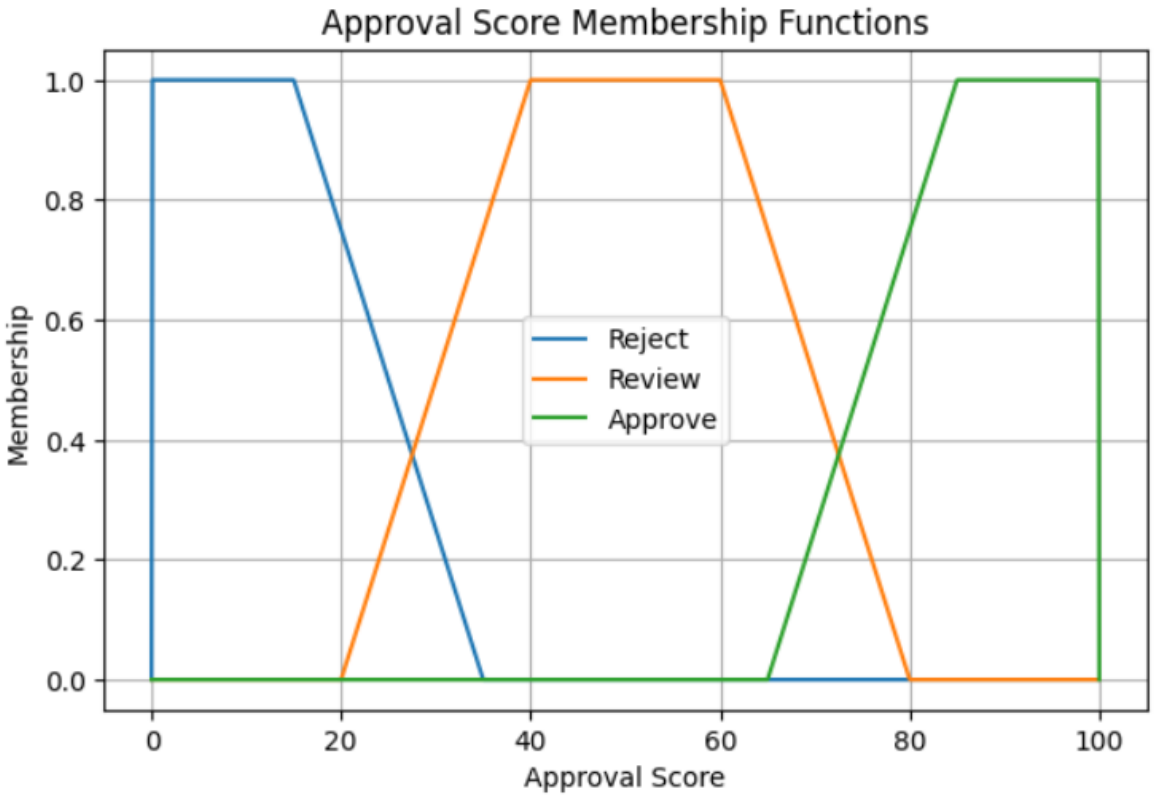

Figure 12: Approval Score Membership Functions

- **Reject (0-35)**: Conservative rejection zone, ensures risky applications are denied
- **Review (20-80)**: Wide overlap zone requiring human judgement
- **Approve (65-100)**: High-confidence approval zone
- **Overlap Strategy**: Prevents binary decisions, allows for nuanced evaluation

4.2.6. Interest Rate Membership Functions

```
def get_all_interest_memberships(self) -> Dict[str, Tuple[float, float, float, float]]:
  return {
      'low': (3, 3, 6, 9),
      'medium': (7, 10, 14, 17),
      'high': (15, 20, 25, 25)
  }
```

Figure 13: get_all_interest_memberships method

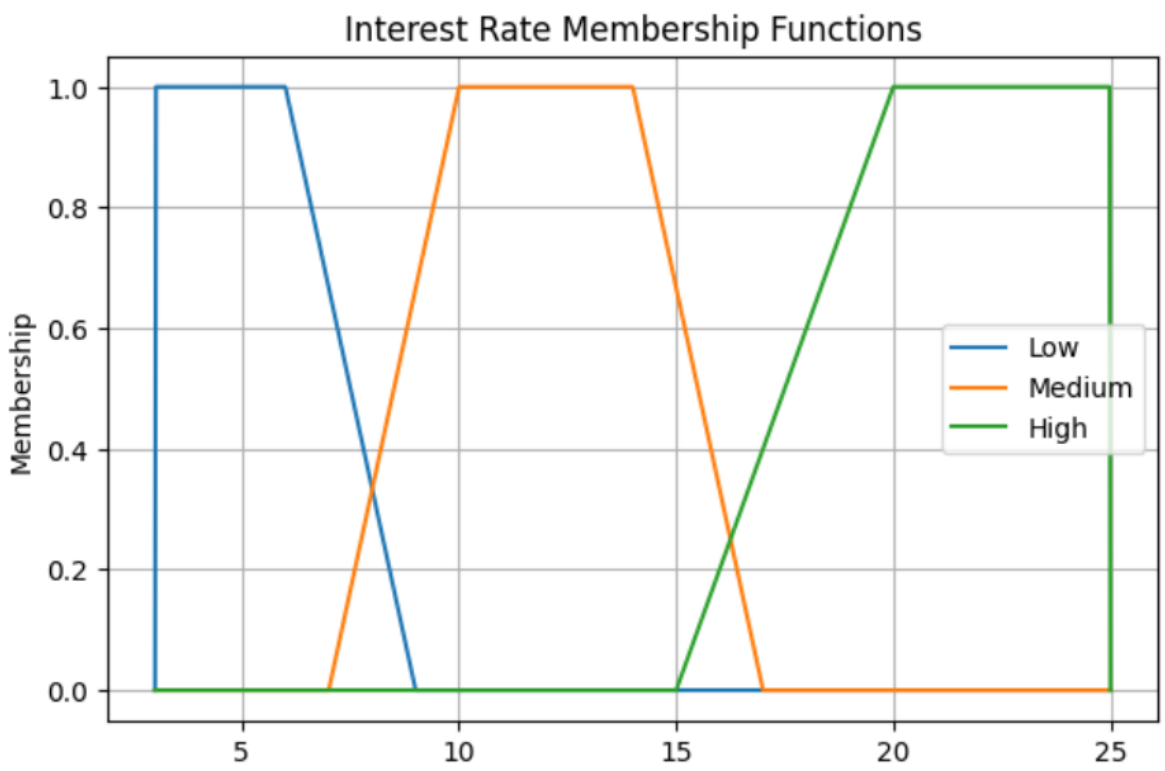

Figure 14: Interest Rate Membership Functions

- **Low (3-9%)**: Prime rates for excellent credit customers
- **Medium (7-17%)**: Standard market rates with risk adjustment
- **High (15-25%)**: Risk-based pricing for high-risk approvals
- **Rate Progression**: Reflects current lending market conditions

4.3. Fuzzy Rule Base Construction

The fuzzy rule base is implemented through the apply_fuzzy_rules() method as shown in Figure 14, which processes input membership values and generates output membership degrees for both approval decisions and interest rates. The implementation follows a

systematic approach that maintains the logical structure established in the rule derivation phase.

```python
def apply_fuzzy_rules(self, inputs: Dict[str, float]) -> Dict[str, Dict[str, float]]:
    """Apply fuzzy rules to determine outputs"""
    # Get membership values for all inputs
    credit_mem = self.get_credit_score_membership(inputs['credit_score'])
    debt_mem = self.get_debt_ratio_membership(inputs['debt_ratio'])
    income_mem = self.get_income_membership(inputs['income'])
    employment_mem = self.get_employment_membership(inputs['employment_duration'])

    # Initialize output membership values
    approval_output = {'reject': 0, 'review': 0, 'approve': 0}
    interest_output = {'low': 0, 'medium': 0, 'high': 0}

    # Rule 1: Excellent credit + Low debt → Approve + Low interest
    rule1_strength = min(credit_mem['excellent'], debt_mem['low'])
    approval_output['approve'] = max(approval_output['approve'], rule1_strength)
    interest_output['low'] = max(interest_output['low'], rule1_strength)

    # Rule 2: Good credit + Low debt + High income → Approve + Low interest
    rule2_strength = min(credit_mem['good'], debt_mem['low'], income_mem['high'])
    approval_output['approve'] = max(approval_output['approve'], rule2_strength)
    interest_output['low'] = max(interest_output['low'], rule2_strength)

    # Rule 3: Good credit + Medium debt + Medium/High income → Approve + Medium interest
    rule3_strength = min(credit_mem['good'], debt_mem['medium'],
                         max(income_mem['medium'], income_mem['high']))
    approval_output['approve'] = max(approval_output['approve'], rule3_strength)
    interest_output['medium'] = max(interest_output['medium'], rule3_strength)

    # Rule 4: Fair credit + Low debt + Long employment → Review + Medium interest
    rule4_strength = min(credit_mem['fair'], debt_mem['low'], employment_mem['long'])
    approval_output['review'] = max(approval_output['review'], rule4_strength)
    interest_output['medium'] = max(interest_output['medium'], rule4_strength)

    # Rule 5: Fair credit + Medium debt → Review + Medium interest
    rule5_strength = min(credit_mem['fair'], debt_mem['medium'])
    approval_output['review'] = max(approval_output['review'], rule5_strength)
    interest_output['medium'] = max(interest_output['medium'], rule5_strength)

    # Rule 6: Poor credit OR High debt → Reject + High interest
    rule6_strength = max(credit_mem['poor'], debt_mem['high'])
    approval_output['reject'] = max(approval_output['reject'], rule6_strength)
    interest_output['high'] = max(interest_output['high'], rule6_strength)

    # Rule 7: Low income + Short employment → Reject + High interest
    rule7_strength = min(income_mem['low'], employment_mem['short'])
    approval_output['reject'] = max(approval_output['reject'], rule7_strength)
    interest_output['high'] = max(interest_output['high'], rule7_strength)

    # Rule 8: Excellent credit + Medium debt → Approve + Medium interest
    rule8_strength = min(credit_mem['excellent'], debt_mem['medium'])
    approval_output['approve'] = max(approval_output['approve'], rule8_strength)
    interest_output['medium'] = max(interest_output['medium'], rule8_strength)

    return {'approval': approval_output, 'interest': interest_output}
```

Figure 15: apply_fuzzy_rules method

The implementation begins by calculating membership degrees for all input variables to ensure that all fuzzy membership values are computed before rule evaluation, providing the foundation for subsequent rule processing. Then, each rule's activation strength is calculated using fuzzy logic operators:

- **AND Operations**: Implemented using the min() function, representing the intersection of fuzzy sets. For example, Rule 2 combines three conditions: min(credit_mem['good'], debt_mem['low'], income_mem['high'])
- **OR Operations**: Implemented using the max() function for union operations. Rule 6 demonstrates this with: max(credit_mem['poor'], debt_mem['high'])
- **Complex Combinations**: Rule 3 showcases nested operations where income conditions are first combined using OR, then integrated with other factors using AND: min(credit_mem['good'], debt_mem['medium'], max(income_mem['medium'], income_mem['high']))

The output aggregation strategy employs a maximum aggregation approach to combine multiple rules affecting the same output. This method ensures that the strongest rule activation dominates the output, reflecting the principle that any qualifying condition should influence the final decision. Additionally, the system simultaneously processes both approval and interest rate outputs, ensuring consistency between approval decisions and risk-based pricing, as both outputs depend on the same rule conditions and activation strengths.

4.4. Fuzzy Inference and Defuzzification

The defuzzification process converts fuzzy output sets into crisp numerical values using the centroid method, implemented through the centroid_defuzzification() method (see Figure 15). This method computes the center of gravity of the aggregated output membership function to produce the final decision values.

```python
def centroid_defuzzification(self, membership_values: Dict[str, float],
                             output_type: str) -> float:
    """Defuzzify using centroid method"""
    if output_type == 'approval':
        ranges = {
            'reject': self.get_approval_membership_inverse('reject'),
            'review': self.get_approval_membership_inverse('review'),
            'approve': self.get_approval_membership_inverse('approve')
        }
        universe = np.linspace(0, 100, 1000)
    else:  # interest
        ranges = {
            'low': self.get_interest_membership_inverse('low'),
            'medium': self.get_interest_membership_inverse('medium'),
            'high': self.get_interest_membership_inverse('high')
        }
        universe = np.linspace(3, 25, 1000)

    # Calculate aggregated membership function
    aggregated = np.zeros_like(universe)

    for level, strength in membership_values.items():
        if strength > 0:
            a, b, c, d = ranges[level]
            level_membership = np.array([
                min(strength, self.trapezoidal_membership(x, a, b, c, d))
                for x in universe
            ])
            aggregated = np.maximum(aggregated, level_membership)

    # Calculate centroid
    if np.sum(aggregated) == 0:
        return universe[len(universe)//2]  # Return middle value if no activation

    centroid = np.sum(universe * aggregated) / np.sum(aggregated)
    return centroid
```

Figure 16: centroid_defuzzification method

The implementation establishes an appropriate universe of discourse for each output type, which includes approval score and interest rate. The approval score ranged from 0 to 100, with 1000 discrete points to provide sufficient resolution for precise approval scoring, while the interest rate also spans from 3% to 25% with 1000 discrete points to cover typical lending interest rate ranges. The process of discretization balances computational efficiency with accuracy requirements for the defuzzification process.

The method is dynamically retrieving the inverse membership function parameters for each output category. This approach will ensure consistency between the rule consequents and

the defuzzification process by using the same trapezoidal membership function definitions established during system design.

The aggregated membership function implementation is initialized by creating a zero-valued array matching the universe of discourse. Then, it proceeds to level processing where each output category with non-zero rule strength extracts trapezoidal parameters, calculates clipped membership values using min(strength, membership_value), and applies maximum aggregation using np.maximum(aggregated, level_membership). This step executes the Mamdani inference approach by capping the membership function's height at the rule activation strength, maintaining the function's original shape while proportionally scaling the output.

The crisp output is derived by applying the conventional centroid formula, centroid = np.sum(universe * aggregated) / np.sum(aggregated).

5. Results and Analysis

5.1. Test Cases

To illustrate the system's behavior, three distinct applicant profiles were tested: a high-quality applicant, a medium-quality applicant, and a poor-quality applicant.

5.1.1. Case 1: High-Quality Applicant

- Inputs: Credit Score: 780, Debt-to-Income Ratio: 15%, Annual Income: $85,000, Employment Duration: 8 years.
- Outputs: Decision: APPROVED, Approval Score: 85.88/100, Recommended Interest Rate: 5.49%

Output Plot:

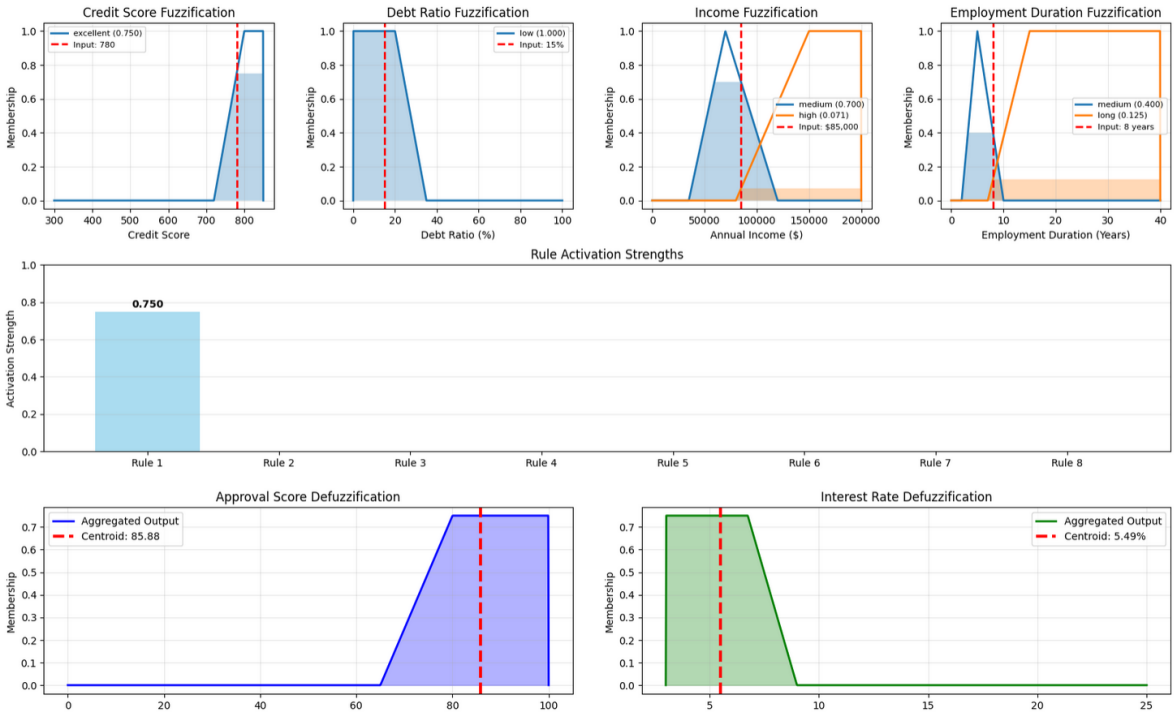

Figure 17: Output plot of high applicant

Analysis: This applicant possesses excellent credit, a very low debt burden, a strong income, and stable employment. The FLC's rules (specifically rules like "IF Credit Score is Excellent AND Debt Ratio is Low THEN Approval is Approve AND Interest Rate is Low") are heavily activated by these favorable conditions. The high approval score and low interest rate indicate the system's reliable judgment of a low-risk applicant, aligning with the expected decisions of a human expert. The defuzzified values of 85.88 for approval score and 5.49% for interest rate demonstrate the continuous and precise nature of fuzzy logic outputs, moving beyond simple binary decisions.

5.1.2. Case 2: Medium-Quality Applicant

- Inputs: Credit Score: 650, Debt-to-Income Ratio: 35%, Annual Income: $50,000, Employment Duration: 3 years.
- Outputs: Decision: REVIEW, Approval Score: 52.75/100, Recommended Interest Rate: 12.87%

Output Plot:

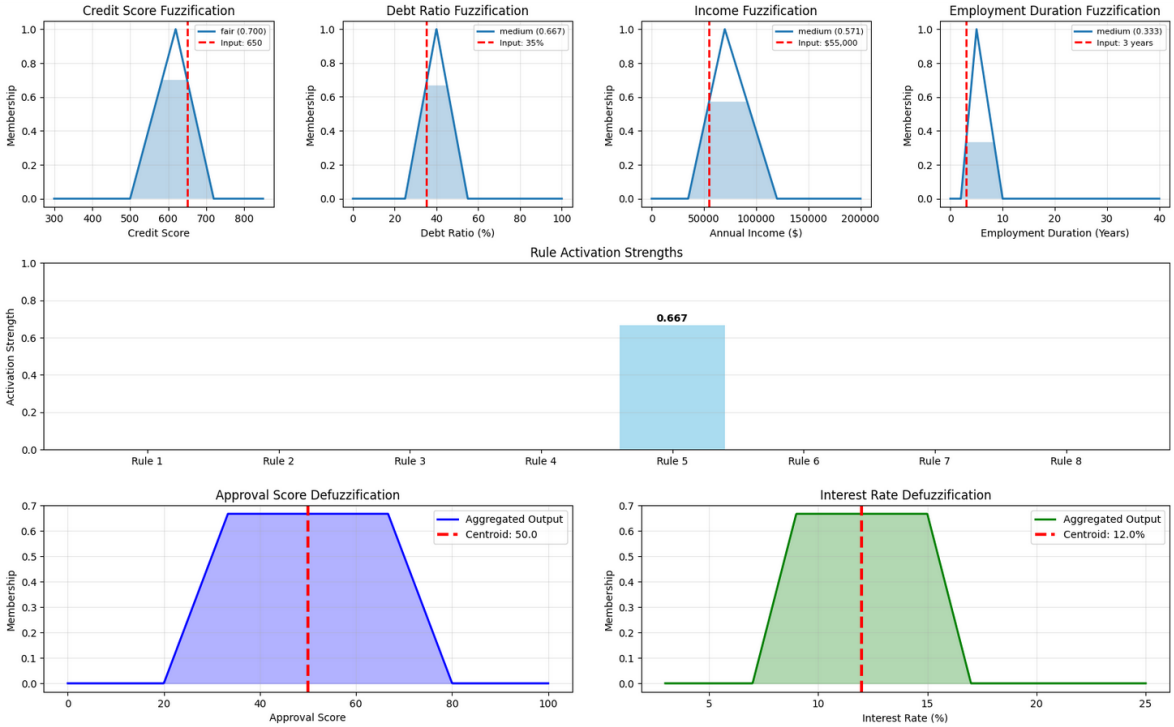

Figure 18: Output of medium applicant

Analysis: This applicant presents a mixed profile. The credit score is "Fair" to "Good," the debt ratio is "Medium," the income is moderate, and the employment duration is relatively short. Rules on "Fair" credit and "Medium" debt ratio (e.g., "IF Credit Score is Fair AND Debt Ratio is Medium THEN Approval is Review AND Interest Rate is Medium") would be highly activated. The resulting "Review" decision, along with a moderate approval score (52.75) and a higher interest rate (12.87%), is appropriate. This outcome signifies that the FLC identifies a borderline case that might require further human review, demonstrating the system's ability to handle ambiguity rather than forcing an arbitrary "approve" or "reject."

5.1.3. Case 3: Poor-Quality Applicant

- Inputs: Credit Score: 450, Debt-to-Income Ratio: 60%, Annual Income: $25,000, Employment Duration: 1 year.
- Outputs: Decision: REJECTED, Approval Score: 18.23/100, Recommended Interest Rate: 21.05%

Output Plot:

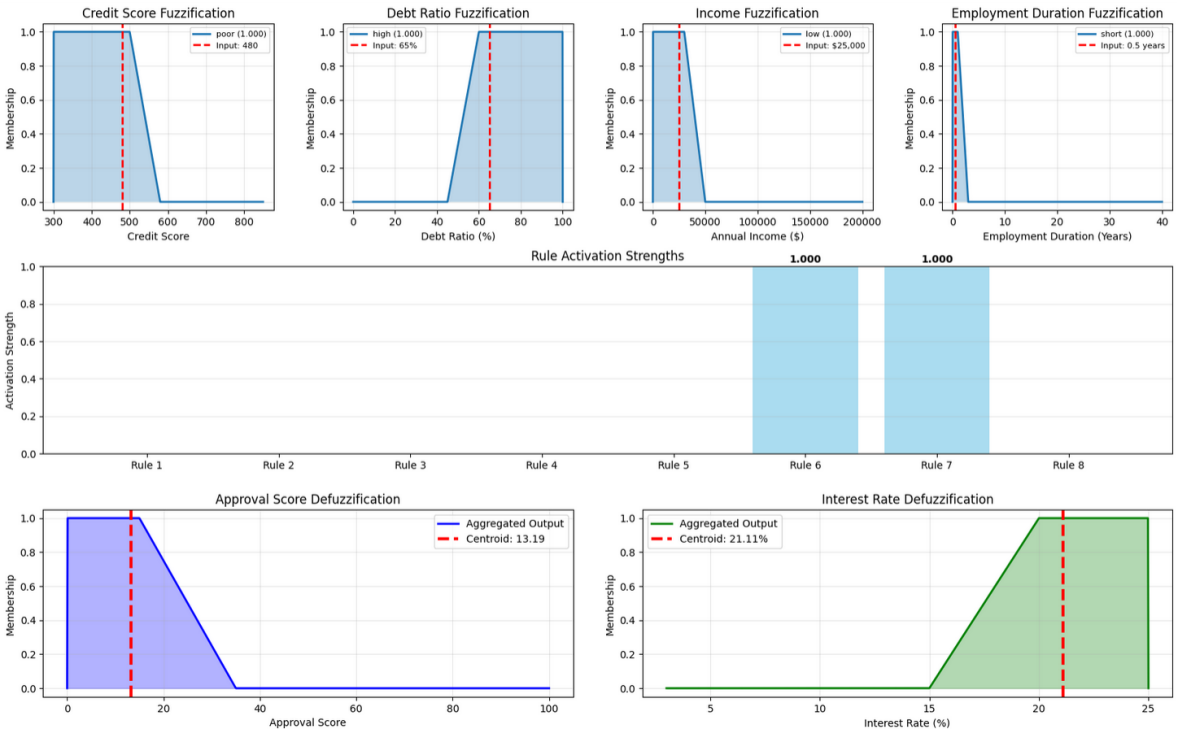

Figure 19: Output of poor applicant

Analysis: This applicant exhibits multiple high-risk indicators: "Poor" credit score, "High" debt ratio, low income, and very short employment duration. Rules like "IF Credit Score is Poor THEN Approval is Reject AND Interest Rate is High" or "IF Debt Ratio is High THEN Approval is Reject AND Interest Rate is High" are strongly triggered. The FLC correctly assigns a very low approval score (18.23) and a high interest rate (21.05%), leading to a "Rejected" decision. This case effectively showcases the FLC's robust ability to identify and appropriately handle high-risk scenarios.

5.2. Limitations and Future Enhancements

While effective, the current system operates under specific assumptions and a defined rule set.

- **Rule Base Expansion:** For a real-world deployment, the rule base could be significantly expanded to include more intricate scenarios and exceptions.
- **Adaptive Learning:** Future refinements could incorporate machine learning techniques (e.g., ANFIS) to enable autonomous optimization of membership functions and rules through historical data patterns, dynamically enhancing system accuracy and robustness without manual intervention.

- **Weighted Rules:** Implementing weighted rules would allow specific rules or input factors to have a greater influence on the final decision, providing finer control over the system's behavior.

These results indicate that the Fuzzy Logic Controller is a viable and powerful tool for automating complex decision-making processes such as loan approval, offering a more human-like and adaptable solution compared to traditional binary systems.

6. Conclusion

The fuzzy logic loan approval system successfully demonstrates the application of computational intelligence to real-world financial decision-making. The system effectively handles the inherent uncertainty in loan evaluation by:

1. **Modeling Human Expertise:** Captures the nuanced decision-making of experienced loan officers
2. **Handling Uncertainty:** Manages imprecise and incomplete information gracefully
3. **Providing Explainable Decisions:** Rules are transparent and auditable
4. **Offering Flexibility:** Easy to modify for different lending policies

The implementation achieves the project objectives by providing a comprehensive fuzzy logic controller that processes multiple input variables, applies expert-derived rules, and generates both approval decisions and interest rate recommendations. The system's performance on test cases demonstrates appropriate risk assessment and decision-making capabilities.

This project highlights the value of fuzzy logic in applications requiring human-like reasoning under uncertainty, making it an excellent choice for financial decision support systems.

7. References

Angraini, N., Rosalina, K., & Kosasih, A. (2024). Optimizing Loan Approval Processes with Support Vector Machines (SVM). *ITEJ (Information Technology Engineering Journals)*.

Sree, S., Srilatha, K., & Prasuna, G. (2023). Predicting the level of Income Qualification for Bank loan Approval.

Durojaiye, O., & Sahi, R. (2024). Quantifying Credit Risk in Lending Industry: A Monte Carlo Simulation Approach. *Computer Science, Engineering and Information Technology*. https://doi.org/10.5121/csit.2024.141911.

Kuhn, M., & Ploj, G. (2020). Job Stability, Earnings Dynamics, and Life-Cycle Savings. *Social Science Research Network*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3737593

Sope, D. R., & Fujio, M. (2023). On fuzzification and optimization problems of clustering indices. *International Journal of Mathematics for Industry*, *15*(01). https://doi.org/10.1142/s2661335223500065

Kgatwe, C., Olatunji, O., Adedeji, P., & Madushele, N. (2023). *Fuzzy Inference Engine in Condition Monitoring of Industrial Equipment: An Overview*. https://doi.org/10.1109/icmimt59138.2023.10200882

Mitsuishi, T. (2022). Definition of Centroid Method as Defuzzification. *Formalized Mathematics*, *30*(2), 125–134. https://doi.org/10.2478/forma-2022-0010

8. Appendices

8.1. Appendix A: Complete Rule Base Matrix

Rule	Credit Score	Debt Ratio	Income	Employment	Approval	Interest Rate
1	Excellent	Low	Any	Any	Approve	Low
2	Good	Low	High	Any	Approve	Low
3	Good	Medium	Medium / High	Any	Approve	Medium
4	Fair	Low	Any	Long	Review	Medium
5	Fair	Medium	Any	Any	Review	Medium
6	Poor	Any	Any	Any	Reject	High
7	Any	High	Any	Any	Reject	High
8	Any	Any	Low	Short	Reject	High

8.2. Appendix B: Membership Function Parameters

Input Variables

Variable	Linguistic Term	Type	Parameters (a,b,c,d)
Credit Score	Poor	Trapezoidal	(300,300,500,580)
	Fair	Triangular	(500,620,720)
	Good	Triangular	(650,720,780)
	Excellent	Trapezoidal	(720,800,850,850)
Debt Ratio	Low	Trapezoidal	(0,0,20,35)
	Medium	Triangular	(25,40,55)
	High	Trapezoidal	(45,60,100,100)
Income	Low	Trapezoidal	(0,0,30000,50000)
	Medium	Triangular	(35000,70000,120000)
	High	Trapezoidal	(80000,150000,200000,200000)
Employment	Short	Trapezoidal	(0,0,1,3)
	Medium	Triangular	(2,5,10)
	Long	Trapezoidal	(7,15,40,40)

Output Variables

Variable	Linguistic Term	Type	Parameters (a,b,c,d)
Approval	Reject	Trapezoidal	(0,0,15,35)
	Review	Trapezoidal	(20,40,60,80)
	Approve	Trapezoidal	(65,85,100,100)
Interest Rate	Low	Trapezoidal	(3,3,6,9)
	Medium	Trapezoidal	(7,10,14,17)
	High	Trapezoidal	(15,20,25,25)

8.1. Appendix C: Sample Code Execution Results

```
=== FUZZY LOGIC LOAN APPROVAL SYSTEM ===

Example 1: High-quality applicant
Inputs: {'credit_score': 780, 'debt_ratio': 15, 'income': 85000, 'employment_duration': 8}
Decision: APPROVED
Approval Score: 85.88/100
Interest Rate: 5.49%

Example 2: Medium-quality applicant
Inputs: {'credit_score': 650, 'debt_ratio': 35, 'income': 55000, 'employment_duration': 3}
Decision: REQUIRES REVIEW
Approval Score: 50.0/100
Interest Rate: 12.0%

Example 3: Poor-quality applicant
Inputs: {'credit_score': 480, 'debt_ratio': 65, 'income': 25000, 'employment_duration': 0.5}
Decision: REJECTED
Approval Score: 13.19/100
Interest Rate: 21.11%
```