AY: 2025-2026
MIDTERM | **AI-ECUE122**
Nov. 2025

M1-S1: Dept. of Electrical Engineering
Teacher: A. Mhamdi
Time Limit: 1h

This document contains **8** pages numbered from **1** to **8**. Upon receiving it, verify completeness. The **3** tasks are independent and can be solved in any order you prefer. The following rules apply:

❶ **A handwritten double-sided A4** sheet is permitted.

❷ **Any electronic material**, except basic calculator, is prohibited.

❸ **Mysterious or unsupported answers** will not receive full credit.

❹ **Round results** to the nearest thousandth *(i.e., the third digit after the decimal point)*.

❺ **Task N°3:** Correct answers earn points as indicated. There is no negative scoring.

Task N°1

⌛ 25mn | (7 points)

Evaluate a **TSK** fuzzy controller for a chemical reactor with:

- **Input 1:** Reactor temperature $T_c = 58.5°C$
- **Input 2:** Temperature rate of change $\dot{T} = 0.75°C/min$
- **Output:** Coolant valve opening percentage

Input 1: Temperature (T_c)

$$\mu_{LT}(T_c) = \begin{cases} 1 & T_c \le 30 \\ \dfrac{50 - T_c}{20} & 30 < T_c < 50 \\ 0 & T_c \ge 50 \end{cases} \quad \mu_{MT}(T_c) = \begin{cases} 0 & T_c \le 40 \\ \dfrac{T_c - 40}{15} & 40 < T_c < 55 \\ \dfrac{70 - T_c}{15} & 55 \le T_c < 70 \\ 0 & T_c \ge 70 \end{cases} \quad \mu_{HT}(T_c) = \begin{cases} 0 & T_c \le 60 \\ \dfrac{T_c - 60}{15} & 60 < T_c < 75 \\ 1 & T_c \ge 75 \end{cases}$$

Input 2: Rate of Change ($\dot{T}$)

$$\mu_{DEC}(\dot{T}) = \begin{cases} 1 & \dot{T} \le -2 \\ \dfrac{-\dot{T} - 1}{2} & -2 < \dot{T} < 0 \\ 0 & \dot{T} \ge 0 \end{cases} \quad \mu_{STAB}(\dot{T}) = \begin{cases} 0 & \dot{T} \le -1 \\ \dfrac{\dot{T} + 1}{1} & -1 < \dot{T} < 0 \\ \dfrac{1 - \dot{T}}{1} & 0 \le \dot{T} < 1 \\ 0 & \dot{T} \ge 1 \end{cases} \quad \mu_{INC}(\dot{T}) = \begin{cases} 0 & \dot{T} \le 0 \\ \dfrac{\dot{T}}{2} & 0 < \dot{T} < 2 \\ 1 & \dot{T} \ge 2 \end{cases}$$

The fuzzy controller is governed by the following rules:

Rule	Condition	Consequent
$\mathfrak{R}_1$	IF LT AND DEC	$y_1 = 0.2T_c + 1.0\dot{T} + 5.0$
$\mathfrak{R}_2$	IF LT AND STAB	$y_2 = 0.3T_c + 0.5\dot{T} + 8.0$
$\mathfrak{R}_3$	IF LT AND INC	$y_3 = 0.5T_c + 2.0\dot{T} + 12.0$
$\mathfrak{R}_4$	IF MT AND DEC	$y_4 = 0.4T_c + 0.8\dot{T} + 10.0$
$\mathfrak{R}_5$	IF MT AND STAB	$y_5 = 0.6T_c + 1.5\dot{T} + 15.0$
$\mathfrak{R}_6$	IF MT AND INC	$y_6 = 0.8T_c + 3.0\dot{T} + 25.0$
$\mathfrak{R}_7$	IF HT AND DEC	$y_7 = 0.7T_c + 2.0\dot{T} + 20.0$
$\mathfrak{R}_8$	IF HT AND STAB	$y_8 = 0.9T_c + 2.5\dot{T} + 30.0$
$\mathfrak{R}_9$	IF HT AND INC	$y_9 = 1.0T_c + 3.5\dot{T} + 40.0$

Fuzzification $(6 \cdot 0.25pt)$

Input T_c:

$$\mu_{LT}(58.5) = 0$$
$$\mu_{MT}(58.5) = \frac{70 - 58.5}{15} = 0.767$$
$$\mu_{HT}(58.5) = 0$$

Input $\dot{T}$:

$$\mu_{DEC}(0.75) = 0$$
$$\mu_{STAB}(0.75) = \frac{1 - 0.75}{1} = 0.25$$
$$\mu_{INC}(0.75) = \frac{0.75}{2} = 0.375$$

Firing Strengths $(9 \cdot 0.25pt)$

$$\mathfrak{R}_1 \longrightarrow w_1 = \min(0, 0) = 0$$
$$\mathfrak{R}_2 \longrightarrow w_2 = \min(0, 0.25) = 0$$
$$\mathfrak{R}_3 \longrightarrow w_3 = \min(0, 0.375) = 0$$
$$\mathfrak{R}_4 \longrightarrow w_4 = \min(0.767, 0) = 0$$
$$\mathfrak{R}_5 \longrightarrow w_5 = \min(0.767, 0.25) = 0.25$$
$$\mathfrak{R}_6 \longrightarrow w_6 = \min(0.767, 0.375) = 0.375$$
$$\mathfrak{R}_7 \longrightarrow w_7 = \min(0, 0) = 0$$
$$\mathfrak{R}_8 \longrightarrow w_8 = \min(0, 0.25) = 0$$
$$\mathfrak{R}_9 \longrightarrow w_9 = \min(0, 0.375) = 0$$

Consequent Evaluation $(9 \cdot 0.25pt)$

$$y_1 = 0.2(58.5) + 1.0(0.75) + 5.0 = 17.45$$
$$y_2 = 0.3(58.5) + 0.5(0.75) + 8.0 = 25.925$$
$$y_3 = 0.5(58.5) + 2.0(0.75) + 12.0 = 42.75$$
$$y_4 = 0.4(58.5) + 0.8(0.75) + 10.0 = 34.0$$
$$y_5 = 0.6(58.5) + 1.5(0.75) + 15.0 = 51.225$$
$$y_6 = 0.8(58.5) + 3.0(0.75) + 25.0 = 74.05$$
$$y_7 = 0.7(58.5) + 2.0(0.75) + 20.0 = 62.45$$
$$y_8 = 0.9(58.5) + 2.5(0.75) + 30.0 = 84.525$$
$$y_9 = 1.0(58.5) + 3.5(0.75) + 40.0 = 101.125$$

Weighted Average Defuzzification $(2 \cdot 0.25pt)$

Numerator

$$\sum_{i=1}^{9} w_i y_i = 40.575$$

Denominator

$$\sum_{i=1}^{9} w_i = 0.625$$

Final Output *(0.5pt)*

$$y_{\text{final}} = \frac{\sum_{i=1}^{9} w_i y_i}{\sum_{i=1}^{9} w_i} = 64.92\%$$

Task N°2

⌛ 15mn | (3 points)

Consider the nonlinear equation:

$$f(x) = x^3 - 2x - 5.$$

Starting from the initial guess $x_0 = 2$, write a `Julia` program[1] that computes:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}, \qquad x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

```julia
f(x)  = x^3 - 2x - 5
fp(x) = 3x^2 - 2

x = 2.0
for n = 1:2
    x = x - f(x)/fp(x)
end

println("After 2 iterations, x = $x")
```


[1] Aim for an efficient implementation.

INSTITUTE OF TECHNOLOGICAL STUDIES OF BIZERTE

AY: 2025-2026	**Full Name:** ....
M1-S1: Dept. of Electrical Engineering	**ID:** ....
MIDTERM | **AI-ECUE122**	**Class:** RAIA1 ....
Nov. 2025	**Room:** ....
Teacher: A. Mhamdi	**Time Limit:** 1h

✂- -

ANSWER SHEET

Task N°3 ⌛ 20mn | (10 points)

(a) (½ point) What year was `Julia` first released to the public?

○ 2010 √ 2012 ○ 2014 ○ 2016

(b) (½ point) Which indexing convention does `Julia` use for arrays?

- ○ 0-based indexing (starts at 0)
- √ 1-based indexing (starts at 1)
- ○ Either 0 or 1 based, user configurable
- ○ Negative indexing only

(c) (½ point) Which of the following is **NOT** a key design goal of `Julia`?

- ○ High performance
- ○ Dynamic typing with optional static typing
- √ Object-oriented programming as the primary paradigm
- ○ Easy interoperability with other languages

(d) (½ point) What is the difference between = and == in `Julia`?

- ○ No difference, they're interchangeable
- √ = is assignment, == is equality comparison
- ○ = is for numbers, == is for strings
- ○ = is equality, == is assignment

(e) (½ point) What is the membership function in fuzzy logic?

- ○ A function that determines the exact category an element belongs to
- ○ A function that converts fuzzy sets to crisp sets
- √ A function that maps input values to a degree of membership in a fuzzy set, ranging from 0 to 1
- ○ A function that measures the distance between two fuzzy sets

DO NOT WRITE ANYTHING HERE

✂----------

(f) (½ point) In `Julia`'s `Fuzzy.jl`, how is a triangular membership function typically defined?

- √ By specifying three parameters: (`a`, `b`, `c`) representing the left, peak, and right points
- ◯ By specifying only the peak value and width
- ◯ By specifying the mean and standard deviation
- ◯ By specifying the minimum and maximum values only

(g) (½ point) What does the fuzzification process accomplish in a fuzzy inference system?

- ◯ It converts fuzzy sets back into single crisp output values
- ◯ It applies fuzzy rules to determine output membership functions
- √ It converts crisp input values into fuzzy sets with membership degrees
- ◯ It measures the overlap between two fuzzy sets

(h) (½ point) In a `Julia` fuzzy system, what happens when the input value falls in the overlap region between two fuzzy sets?

- √ The input belongs to both fuzzy sets with non-zero membership degrees, with different membership values for each set
- ◯ The input must be assigned to exactly one fuzzy set only
- ◯ The system treats it as an error condition
- ◯ The overlapping region is automatically removed by the system

(i) (½ point) Which of the following best describes `Julia`'s type system?

- ◯ Statically typed like `Java` or C++
- ◯ Untyped like `Python`
- ◯ Requires all variables to be explicitly typed at declaration
- √ Dynamic typing with optional type annotations for performance optimization

(j) (½ point) In `Julia`, what is the primary advantage of multiple dispatch?

- ◯ It enables parallel execution of multiple threads
- √ It allows functions to have different implementations based on the types of all arguments
- ◯ It automatically converts types to match function signatures
- ◯ It eliminates the need for type checking in functions

DO NOT WRITE ANYTHING HERE

✂----------

(k) (½ point) What is the purpose of `Julia`'s JIT (Just-In-Time) compiler?

- √ To compile `Julia` code to machine code at runtime for improved execution speed
- ◯ To interpret code line by line without any compilation
- ◯ To convert `Julia` code to Python code automatically
- ◯ To pre-compile all code before the program starts

(l) (½ point) In `Julia`, what does the `@` symbol represent when used as a prefix (e.g., `@time`, `@elapsed`)?

- ◯ It marks a variable as global scope
- ◯ It indicates a string literal
- √ It denotes a macro that performs compile-time or runtime code transformation
- ◯ It denotes a function call with special priority

(m) (½ point) Which `Julia` data structure is most efficient for numerical computations and matrix operations?

√ Arrays ◯ Dictionaries ◯ Tuples ◯ Sets

(n) (½ point) In `Julia`, what is the difference between a function defined with `function` and an arrow function `x -> y`?

- √ Both define functions; arrow functions are concise one-liners while `function` is for multi-line definitions, but both support multiple dispatch
- ◯ Arrow functions are faster than regular functions
- ◯ Regular functions support multiple dispatch but arrow functions do not
- ◯ Arrow functions can only be used for anonymous functions

(o) (½ point) What is the primary purpose of `Julia`'s package manager `Pkg`?

- ◯ To compile `Julia` code into executable binaries
- ◯ To optimize code performance automatically
- √ To manage dependencies, install packages, and handle version control for `Julia` projects
- ◯ To convert `Julia` packages to C libraries

(p) (½ point) In `Julia`, which of the following is true about vectorized operations *(e.g., `A .* B`)*?

- ◯ They require explicit loops to work correctly

- ◯ They only work with one-dimensional arrays
- ◯ They are slower than explicit loops in `Julia`
- √ They apply element-wise operations to arrays efficiently and are fundamental to `Julia`'s performance

(q) (½ point) What does the broadcasting mechanism in `Julia` allow you to do?

- ◯ To execute code simultaneously across multiple CPU cores
- √ To automatically extend scalar functions to work on arrays by applying them element-wise
- ◯ To convert arrays to distributed arrays across a cluster
- ◯ To share variables between different function scopes

(r) (½ point) What is the significance of `Julia`'s `nothing` value?

- √ It represents the absence of a value or is returned when no explicit return value is provided
- ◯ It is equivalent to `0` in numerical computations
- ◯ It indicates an undefined variable
- ◯ It represents an empty array

(s) (½ point) In `Julia`, what does the `::` operator do?

- ◯ It creates a reference to a variable
- ◯ It defines a new type
- √ It provides type annotation to assert or declare the expected type of a variable or argument
- ◯ It performs type conversion automatically

(t) (½ point) Which of the following best describes `Julia`'s performance model?

- ◯ `Julia` is designed primarily for rapid prototyping with performance as a secondary concern
- ◯ `Julia` cannot achieve performance competitive with compiled languages
- ◯ `Julia` requires manual memory management like C++
- √ `Julia` approaches C/`Fortran`-level performance through type specialization and JIT compilation