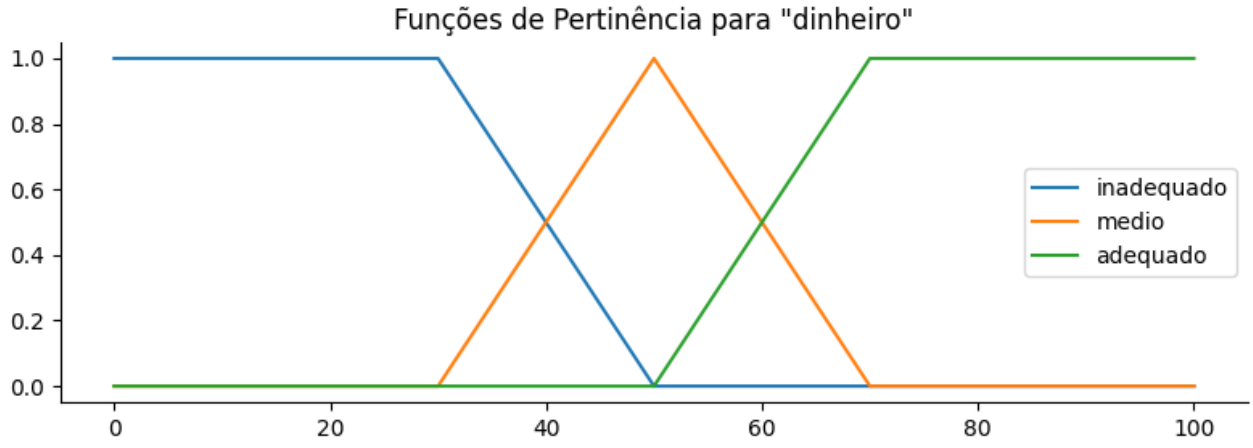
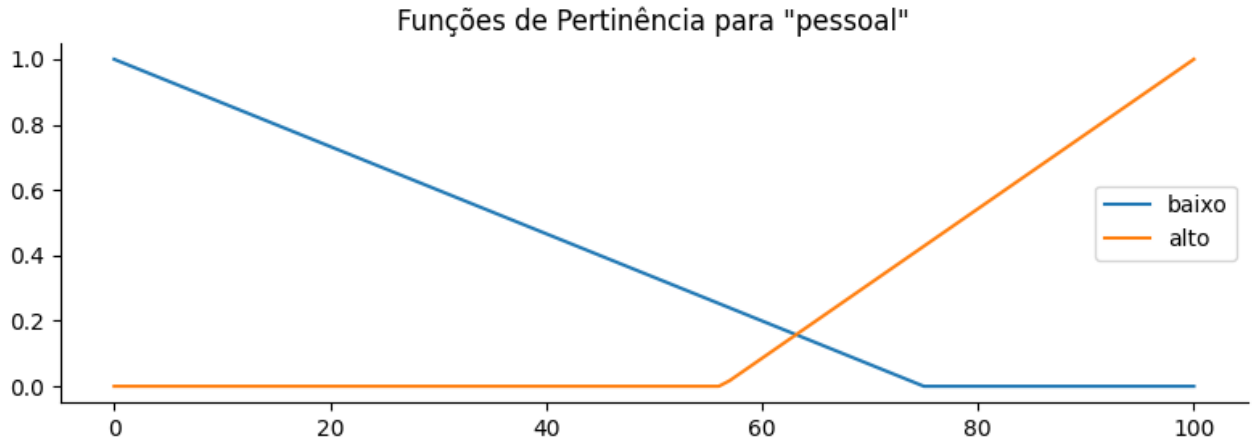
Relatório do Trabalho Prático de IA - Lógica Fuzzy

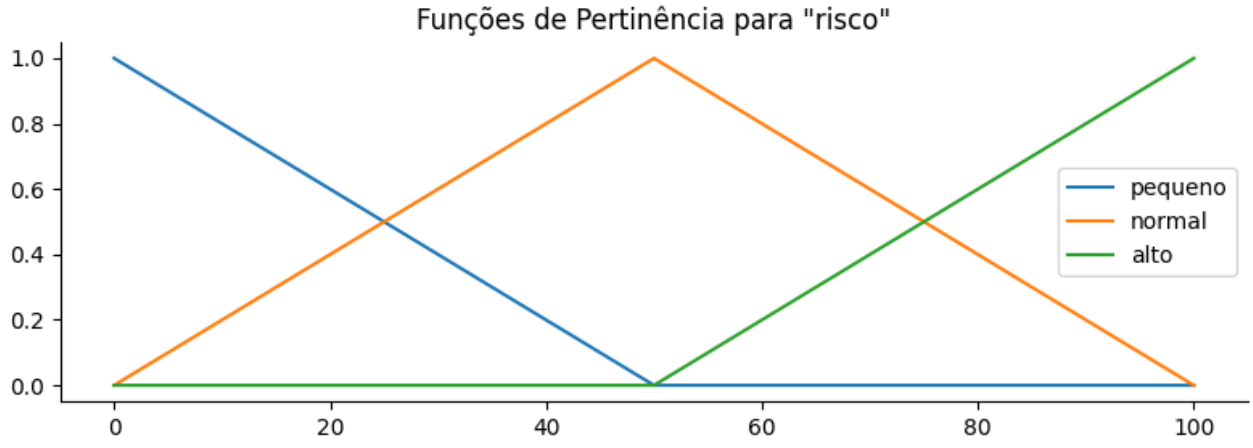
Autor: Pedro Henrique Vilaça Valverde

Exercício 1: Análise de Risco de Projeto

Funções de Pertinência:







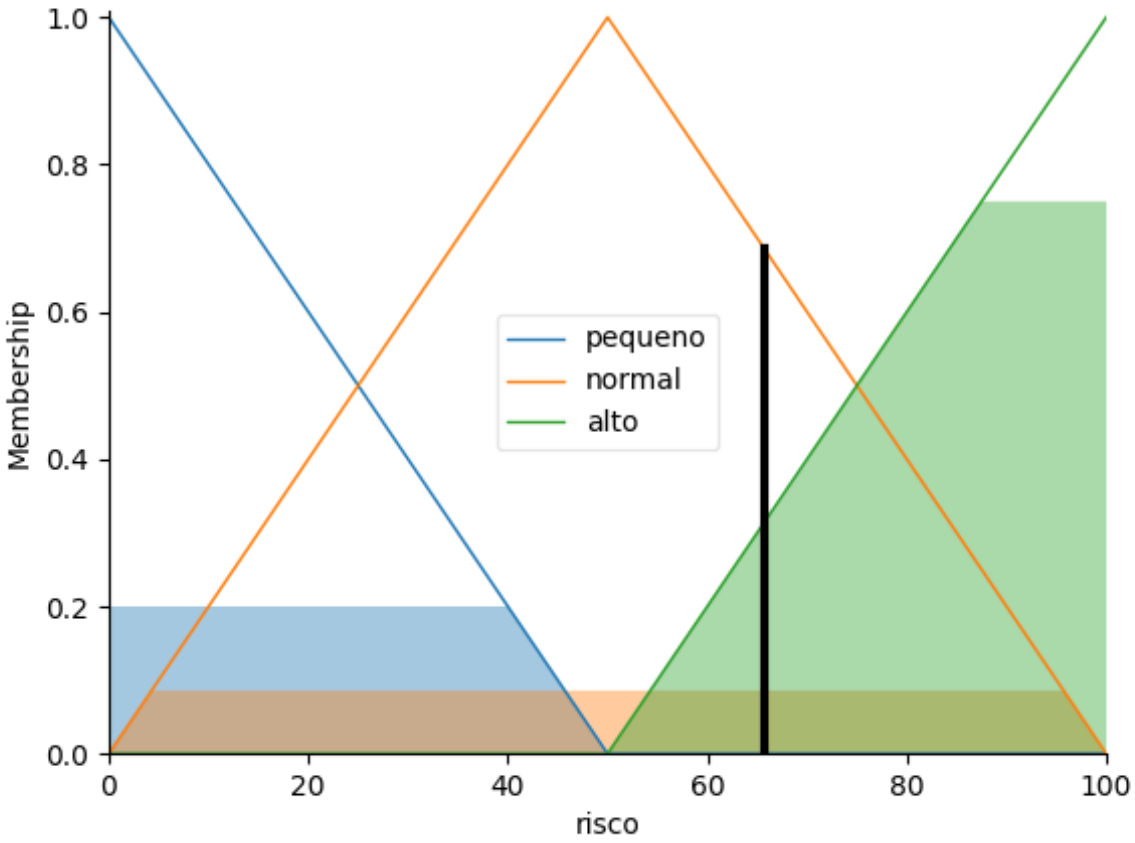
Base de Conhecimento (Regras):

```
1. Se dinheiro é 'adequado' OU pessoal é 'baixo' ENTÃO risco é 'pequeno'
2. Se dinheiro é 'medio' E pessoal é 'alto' ENTÃO risco é 'normal'
3. Se dinheiro é 'inadequado' ENTÃO risco é 'alto'
```

Resultado da Simulação:

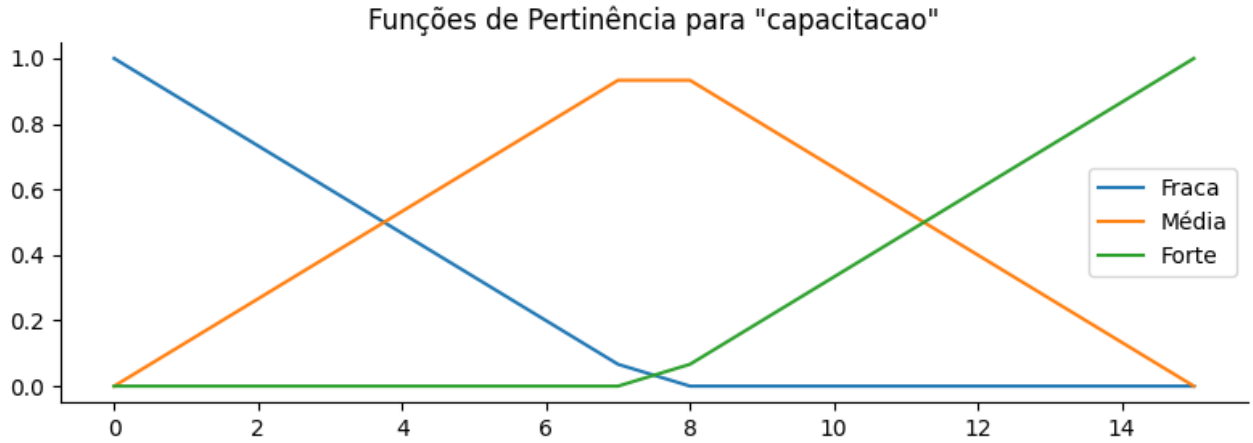
```
Entradas: Dinheiro = 35%, Pessoal = 60%
Valor do Risco (Defuzzificado): 65.70
```

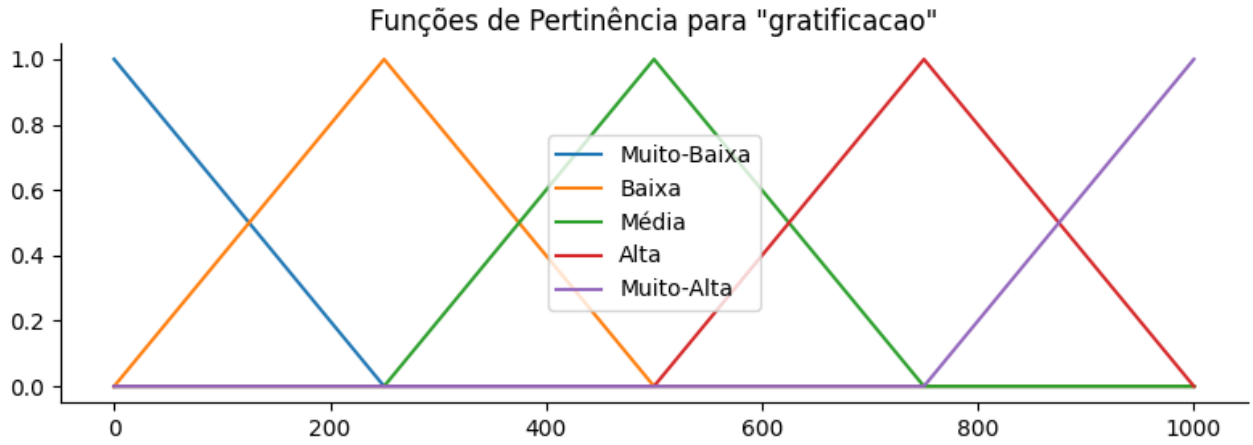
Gráfico de Saída (Defuzzificação):



Exercício 2: Sistema de Gratificação de RH

Funções de Pertinência:



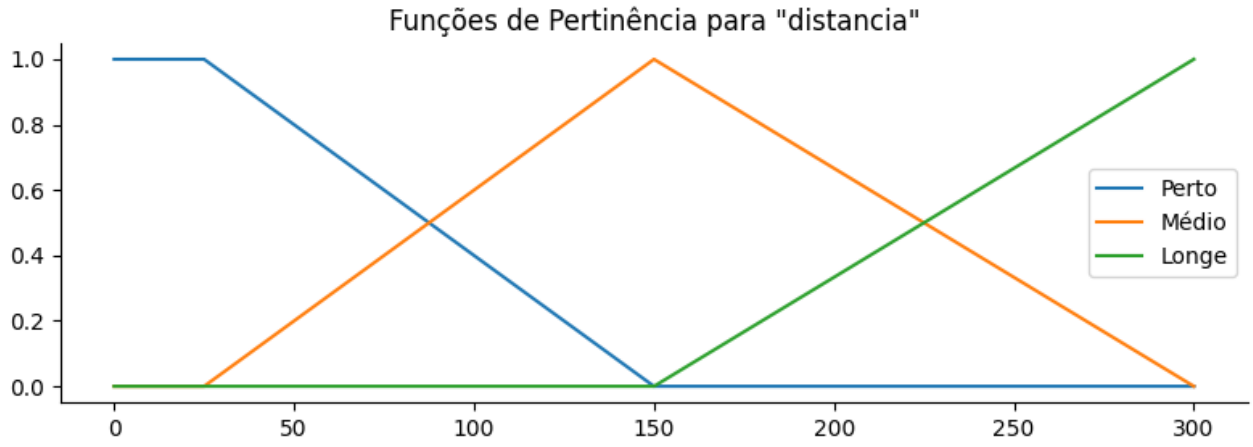


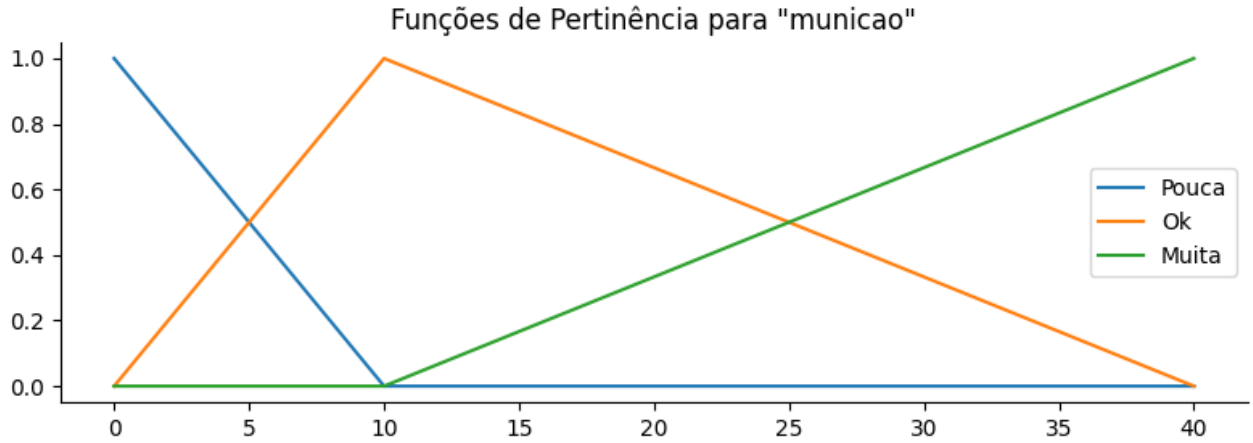
Resultados para os Cenários Propostos:

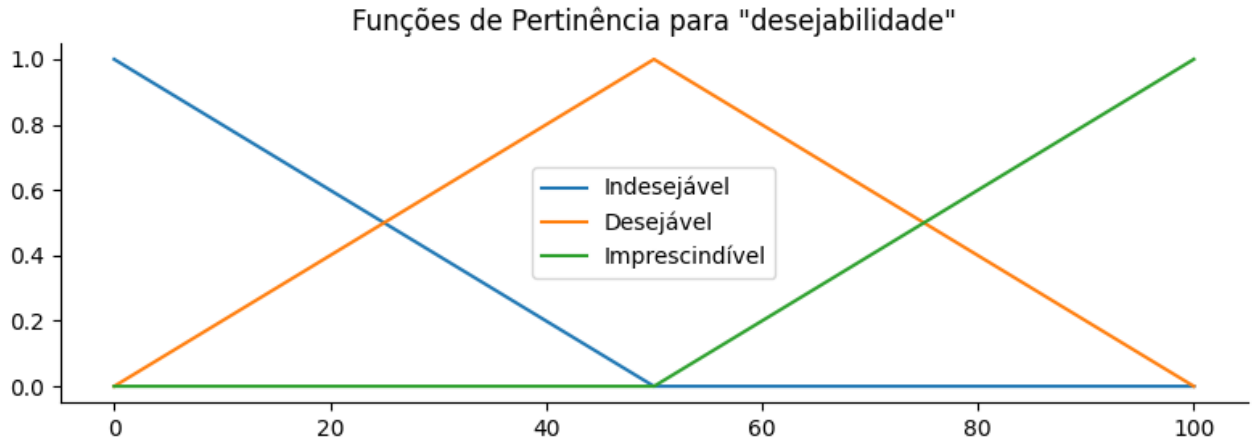
```
Cenário a) Capacitação: 10 anos, Experiência: 20 anos -> Gratificação: R$ 607.49
Cenário b) Capacitação: 14 anos, Experiência: 27 anos -> Gratificação: R$ 763.39
Cenário c) Capacitação: 2 anos, Experiência: 5 anos -> Gratificação: R$ 295.86
Cenário d) Capacitação: 5 anos, Experiência: 26 anos -> Gratificação: R$ 569.40
Cenário e) Capacitação: 8 anos, Experiência: 18 anos -> Gratificação: R$ 561.18
```

Exercício 3: IA para Seleção de Armas em Jogo

Funções de Pertinência:







Tabelas de Regras (Lógica Definida):

```
--- Tabela de Regras para: Lançador de Foguetes ---
Munição    | Perto           | Médio           | Longe
-----------------------------------------------------------
Pouca      | Indesejável     | Desejável       | Indesejável
Ok         | Indesejável     | Imprescindível  | Indesejável
Muita      | Indesejável     | Imprescindível  | Desejável

--- Tabela de Regras para: Revolver ---
```

```
Munição    | Perto           | Médio           | Longe
-------------------------------------------------------------
Pouca      | Desejável       | Indesejável     | Indesejável
Ok         | Imprescindível  | Desejável       | Indesejável
Muita      | Imprescindível  | Desejável       | Indesejável

--- Tabela de Regras para: Sniper ---
Munição    | Perto           | Médio           | Longe
-------------------------------------------------------------
Pouca      | Indesejável     | Indesejável     | Desejável
Ok         | Indesejável     | Desejável       | Imprescindível
Muita      | Indesejável     | Desejável       | Imprescindível
```

Análise e Resultados dos Cenários:

```
--- RESULTADOS PARA CENÁRIO 1 (Distância=200, Munição=8) ---
  - Desejabilidade para 'Lançador de Foguetes': 58.10
  - Desejabilidade para 'Revolver': 47.38
  - Desejabilidade para 'Sniper': 51.59
==> MELHOR ESCOLHA: 'Lançador de Foguetes' (Score: 58.10)

--- RESULTADOS PARA CENÁRIO 2 (Distância=270, Munição=8) ---
  - Desejabilidade para 'Lançador de Foguetes': 34.57
  - Desejabilidade para 'Revolver': 32.75
  - Desejabilidade para 'Sniper': 65.43
==> MELHOR ESCOLHA: 'Sniper' (Score: 65.43)
```

Código Fonte: resolver_fuzzy.py

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
import os

# --- FUNÇÃO AUXILIAR PARA SALVAR GRÁFICOS ---
def save_variable_plot(variable, file_name):
    """
    Plota e salva o gráfico de uma variável fuzzy (Antecedent/Consequent)
    em um arquivo, pois variable.view() não permite salvar diretamente.
    """
    fig, ax = plt.subplots(figsize=(8, 3))
    for term in variable.terms:
        ax.plot(variable.universe, variable[term].mf, label=term)
    ax.set_title(f'Funções de Pertinência para "{variable.label}"')
    ax.legend()
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
    plt.tight_layout()
    plt.savefig(file_name)
    plt.close(fig)

# --- CRIAÇÃO DO DIRETÓRIO DE RESULTADOS ---
RESULT_DIR = './result'
if not os.path.exists(RESULT_DIR):
    os.makedirs(RESULT_DIR)
print(f"Diretório '{RESULT_DIR}' pronto para receber os resultados.")

# --- EXERCÍCIO 1: ANÁLISE DE RISCO DE PROJETO ---
def solve_problem_1():
    print("\n--- Iniciando Exercício 1: Análise de Risco ---")

    # 1. Definição das variáveis (Universos)
    dinheiro = ctrl.Antecedent(np.arange(0, 101, 1), 'dinheiro')
    pessoal = ctrl.Antecedent(np.arange(0, 101, 1), 'pessoal')
    risco = ctrl.Consequent(np.arange(0, 101, 1), 'risco')

    # 2. Definição das Funções de Pertinência (Fuzzificação)
    dinheiro['inadequado'] = fuzz.trapmf(dinheiro.universe, [0, 0, 30, 50])
    dinheiro['medio'] = fuzz.trimf(dinheiro.universe, [30, 50, 70])
    dinheiro['adequado'] = fuzz.trapmf(dinheiro.universe, [50, 70, 100, 100])

    pessoal['baixo'] = fuzz.trimf(pessoal.universe, [0, 0, 75])
    pessoal['alto'] = fuzz.trimf(pessoal.universe, [56.25, 100, 100])

    risco['pequeno'] = fuzz.trimf(risco.universe, [0, 0, 50])
    risco['normal'] = fuzz.trimf(risco.universe, [0, 50, 100])
    risco['alto'] = fuzz.trimf(risco.universe, [50, 100, 100])

    save_variable_plot(dinheiro, os.path.join(RESULT_DIR, 'ex1_dinheiro_mf.png'))
    save_variable_plot(pessoal, os.path.join(RESULT_DIR, 'ex1_pessoal_mf.png'))
    save_variable_plot(risco, os.path.join(RESULT_DIR, 'ex1_risco_mf.png'))
    print("Gráficos das funções de pertinência do Exercício 1 salvos.")
```

```
    # 3. Base de Conhecimento (Regras)
    regra1 = ctrl.Rule(dinheiro['adequado'] | pessoal['baixo'], risco['pequeno'])
    regra2 = ctrl.Rule(dinheiro['medio'] & pessoal['alto'], risco['normal'])
    regra3 = ctrl.Rule(dinheiro['inadequado'], risco['alto'])

    with open(os.path.join(RESULT_DIR, 'ex1_regras.txt'), 'w', encoding='utf-8') as f:
        f.write("1. Se dinheiro é 'adequado' OU pessoal é 'baixo' ENTÃO risco é 'pequeno'\n")
        f.write("2. Se dinheiro é 'medio' E pessoal é 'alto' ENTÃO risco é 'normal'\n")
        f.write("3. Se dinheiro é 'inadequado' ENTÃO risco é 'alto'\n")

    # 4. Sistema de Inferência e Simulação
    sistema_risco_ctrl = ctrl.ControlSystem([regra1, regra2, regra3])
    simulador_risco = ctrl.ControlSystemSimulation(sistema_risco_ctrl)

    simulador_risco.input['dinheiro'] = 35
    simulador_risco.input['pessoal'] = 60
    simulador_risco.compute()
    resultado_risco = simulador_risco.output['risco']

    print(f"Resultado (Risco) para Dinheiro=35 e Pessoal=60: {resultado_risco:.2f}")

    with open(os.path.join(RESULT_DIR, 'ex1_resultado.txt'), 'w', encoding='utf-8') as f:
        f.write(f"Entradas: Dinheiro = 35%, Pessoal = 60%\n")
        f.write(f"Valor do Risco (Defuzzificado): {resultado_risco:.2f}\n")

    risco.view(sim=simulador_risco)
    plt.savefig(os.path.join(RESULT_DIR, 'ex1_risco_resultado_grafico.png'))
    plt.close()
    print("Gráfico de resultado do Exercício 1 salvo.")
    print("--- Exercício 1 Finalizado ---")

# --- EXERCÍCIO 2: SISTEMA DE GRATIFICAÇÃO ---
def solve_problem_2():
    print("\n--- Iniciando Exercício 2: Sistema de Gratificação ---")

    # 1. Definição das variáveis
    experiencia = ctrl.Antecedent(np.arange(0, 31, 1), 'experiencia')
    capacitacao = ctrl.Antecedent(np.arange(0, 16, 1), 'capacitacao')
    gratificacao = ctrl.Consequent(np.arange(0, 1001, 1), 'gratificacao')

    # 2. Funções de Pertinência
    experiencia['Pouca'] = fuzz.trimf(experiencia.universe, [0, 0, 15])
    experiencia['Média'] = fuzz.trimf(experiencia.universe, [0, 15, 30])
    experiencia['Muita'] = fuzz.trimf(experiencia.universe, [15, 30, 30])

    capacitacao['Fraca'] = fuzz.trimf(capacitacao.universe, [0, 0, 7.5])
    capacitacao['Média'] = fuzz.trimf(capacitacao.universe, [0, 7.5, 15])
    capacitacao['Forte'] = fuzz.trimf(capacitacao.universe, [7.5, 15, 15])

    gratificacao['Muito-Baixa'] = fuzz.trimf(gratificacao.universe, [0, 0, 250])
    gratificacao['Baixa'] = fuzz.trimf(gratificacao.universe, [0, 250, 500])
    gratificacao['Média'] = fuzz.trimf(gratificacao.universe, [250, 500, 750])
    gratificacao['Alta'] = fuzz.trimf(gratificacao.universe, [500, 750, 1000])
    gratificacao['Muito-Alta'] = fuzz.trimf(gratificacao.universe, [750, 1000, 1000])

    save_variable_plot(experiencia, os.path.join(RESULT_DIR, 'ex2_experiencia_mf.png'))
    save_variable_plot(capacitacao, os.path.join(RESULT_DIR, 'ex2_capacitacao_mf.png'))
    save_variable_plot(gratificacao, os.path.join(RESULT_DIR, 'ex2_gratificacao_mf.png'))
    print("Gráficos das funções de pertinência do Exercício 2 salvos.")
```

```
    # 3. Regras
    regras_texto = []
    regras = [
        ctrl.Rule(capacitacao['Fraca'] & experiencia['Pouca'], gratificacao['Muito-Baixa']),
        ctrl.Rule(capacitacao['Fraca'] & experiencia['Média'], gratificacao['Baixa']),
        ctrl.Rule(capacitacao['Fraca'] & experiencia['Muita'], gratificacao['Média']),
        ctrl.Rule(capacitacao['Média'] & experiencia['Pouca'], gratificacao['Baixa']),
        ctrl.Rule(capacitacao['Média'] & experiencia['Média'], gratificacao['Média']),
        ctrl.Rule(capacitacao['Média'] & experiencia['Muita'], gratificacao['Alta']),
        ctrl.Rule(capacitacao['Forte'] & experiencia['Pouca'], gratificacao['Média']),
        ctrl.Rule(capacitacao['Forte'] & experiencia['Média'], gratificacao['Alta']),
        ctrl.Rule(capacitacao['Forte'] & experiencia['Muita'], gratificacao['Muito-Alta'])
    ]
    for i, r in enumerate(regras):
        regras_texto.append(f"{i+1}. {r.antecedent} => {r.consequent}")

    with open(os.path.join(RESULT_DIR, 'ex2_regras.txt'), 'w', encoding='utf-8') as f:
        f.write('\n'.join(regras_texto))

    # 4. Sistema e Simulação
    gratificacao_ctrl = ctrl.ControlSystem(regras)
    simulador_gratificacao = ctrl.ControlSystemSimulation(gratificacao_ctrl)

    # 5. Cenários
    cenarios = [
        {'cap': 10, 'exp': 20, 'label': 'a'},
        {'cap': 14, 'exp': 27, 'label': 'b'},
        {'cap': 2, 'exp': 5, 'label': 'c'},
        {'cap': 5, 'exp': 26, 'label': 'd'},
        {'cap': 8, 'exp': 18, 'label': 'e'}
    ]

    resultados_finais = []
    print("Calculando gratificações para os cenários propostos...")
    for cenario in cenarios:
        simulador_gratificacao.input['capacitacao'] = cenario['cap']
        simulador_gratificacao.input['experiencia'] = cenario['exp']
        simulador_gratificacao.compute()
        resultado = simulador_gratificacao.output['gratificacao']
        linha = f"Cenário {cenario['label']}) Capacitação: {cenario['cap']} anos, Experiência: {cenario['exp']}
anos -> Gratificação: R$ {resultado:.2f}"
        resultados_finais.append(linha)
        print(linha)

    with open(os.path.join(RESULT_DIR, 'ex2_resultados.txt'), 'w', encoding='utf-8') as f:
        f.write('\n'.join(resultados_finais))

    print("--- Exercício 2 Finalizado ---")

# --- EXERCÍCIO 3: IA PARA SELEÇÃO DE ARMAS ---
def solve_problem_3():
    print("\n--- Iniciando Exercício 3: IA para Seleção de Armas ---")

    # 1. Variáveis
    distancia = ctrl.Antecedent(np.arange(0, 301, 1), 'distancia')
    municao = ctrl.Antecedent(np.arange(0, 41, 1), 'municao')
    desejabilidade = ctrl.Consequent(np.arange(0, 101, 1), 'desejabilidade')

    # 2. Funções de Pertinência
    distancia['Perto'] = fuzz.trapmf(distancia.universe, [0, 0, 25, 150])
```

```
distancia['Médio'] = fuzz.trimf(distancia.universe, [25, 150, 300])
distancia['Longe'] = fuzz.trapmf(distancia.universe, [150, 300, 300, 300])

municao['Pouca'] = fuzz.trimf(municao.universe, [0, 0, 10])
municao['Ok'] = fuzz.trimf(municao.universe, [0, 10, 40])
municao['Muita'] = fuzz.trimf(municao.universe, [10, 40, 40])

desejabilidade['Indesejável'] = fuzz.trimf(desejabilidade.universe, [0, 0, 50])
desejabilidade['Desejável'] = fuzz.trimf(desejabilidade.universe, [0, 50, 100])
desejabilidade['Imprescindível'] = fuzz.trimf(desejabilidade.universe, [50, 100, 100])

save_variable_plot(distancia, os.path.join(RESULT_DIR, 'ex3_distancia_mf.png'))
save_variable_plot(municao, os.path.join(RESULT_DIR, 'ex3_municao_mf.png'))
save_variable_plot(desejabilidade, os.path.join(RESULT_DIR, 'ex3_desejabilidade_mf.png'))
print("Gráficos das funções de pertinência do Exercício 3 salvos.")

# 3. Regras por Arma
regras = {
    'Lançador de Foguetes': [
        ctrl.Rule(municao['Pouca'] & distancia['Perto'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Pouca'] & distancia['Médio'], desejabilidade['Desejável']),
        ctrl.Rule(municao['Pouca'] & distancia['Longe'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Ok'] & distancia['Perto'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Ok'] & distancia['Médio'], desejabilidade['Imprescindível']),
        ctrl.Rule(municao['Ok'] & distancia['Longe'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Muita'] & distancia['Perto'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Muita'] & distancia['Médio'], desejabilidade['Imprescindível']),
        ctrl.Rule(municao['Muita'] & distancia['Longe'], desejabilidade['Desejável']),
    ],
    'Revolver': [
        ctrl.Rule(municao['Pouca'] & distancia['Perto'], desejabilidade['Desejável']),
        ctrl.Rule(municao['Pouca'] & distancia['Médio'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Pouca'] & distancia['Longe'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Ok'] & distancia['Perto'], desejabilidade['Imprescindível']),
        ctrl.Rule(municao['Ok'] & distancia['Médio'], desejabilidade['Desejável']),
        ctrl.Rule(municao['Ok'] & distancia['Longe'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Muita'] & distancia['Perto'], desejabilidade['Imprescindível']),
        ctrl.Rule(municao['Muita'] & distancia['Médio'], desejabilidade['Desejável']),
        ctrl.Rule(municao['Muita'] & distancia['Longe'], desejabilidade['Indesejável']),
    ],
    'Sniper': [
        ctrl.Rule(municao['Pouca'] & distancia['Perto'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Pouca'] & distancia['Médio'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Pouca'] & distancia['Longe'], desejabilidade['Desejável']),
        ctrl.Rule(municao['Ok'] & distancia['Perto'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Ok'] & distancia['Médio'], desejabilidade['Desejável']),
        ctrl.Rule(municao['Ok'] & distancia['Longe'], desejabilidade['Imprescindível']),
        ctrl.Rule(municao['Muita'] & distancia['Perto'], desejabilidade['Indesejável']),
        ctrl.Rule(municao['Muita'] & distancia['Médio'], desejabilidade['Desejável']),
        ctrl.Rule(municao['Muita'] & distancia['Longe'], desejabilidade['Imprescindível']),
    ]
}

# Salvando as tabelas de regras em texto
with open(os.path.join(RESULT_DIR, 'ex3_regras_tabelas.txt'), 'w', encoding='utf-8') as f:
    d_labels = ['Perto', 'Médio', 'Longe']
    m_labels = ['Pouca', 'Ok', 'Muita']
    for arma, lista_regras in regras.items():
        f.write(f"--- Tabela de Regras para: {arma} ---\n")
        f.write(f"{'Munição':<10} | {'Perto':<15} | {'Médio':<15} | {'Longe':<15}\n")
```

```
            f.write("-" * 60 + "\n")

            tabela = {m: {d: 'N/A' for d in d_labels} for m in m_labels}
            for r in lista_regras:
                antecedent_terms = [r.antecedent.term1, r.antecedent.term2]
                mun_term = next(t for t in antecedent_terms if t.parent.label == 'municao').label
                dist_term = next(t for t in antecedent_terms if t.parent.label == 'distancia').label
                des_term = r.consequent[0].term.label
                tabela[mun_term][dist_term] = des_term

            for m in m_labels:
                                f.write(f"{m:<10}  |  {tabela[m]['Perto']:<15}  |  {tabela[m]['Médio']:<15}  |
{tabela[m]['Longe']:<15}\n")
            f.write("\n")

    # 4. Simulação dos Cenários
    cenarios = [
        {'dist': 200, 'mun': 8, 'label': '1'},
        {'dist': 270, 'mun': 8, 'label': '2'}
    ]

    resultados_finais = []
    for cenario in cenarios:
                            print(f"\nAnalisando    Cenário    {cenario['label']}:    Distância={cenario['dist']},
Munição={cenario['mun']}")
                            resultados_finais.append(f"\n---    RESULTADOS    PARA    CENÁRIO    {cenario['label']}
(Distância={cenario['dist']}, Munição={cenario['mun']}) ---")

        scores = {}
        for arma, regras_arma in regras.items():
            sistema_ctrl = ctrl.ControlSystem(regras_arma)
            simulador = ctrl.ControlSystemSimulation(sistema_ctrl)
            simulador.input['distancia'] = cenario['dist']
            simulador.input['municao'] = cenario['mun']
            simulador.compute()
            scores[arma] = simulador.output['desejabilidade']
            linha = f"  - Desejabilidade para '{arma}': {scores[arma]:.2f}"
            print(linha)
            resultados_finais.append(linha)

        melhor_arma = max(scores, key=scores.get)
        linha_final = f"==> MELHOR ESCOLHA: '{melhor_arma}' (Score: {scores[melhor_arma]:.2f})"
        print(linha_final)
        resultados_finais.append(linha_final)

    with open(os.path.join(RESULT_DIR, 'ex3_resultados.txt'), 'w', encoding='utf-8') as f:
        f.write('\n'.join(resultados_finais))

    print("--- Exercício 3 Finalizado ---")

# --- EXECUÇÃO PRINCIPAL ---
if __name__ == "__main__":
    solve_problem_1()
    solve_problem_2()
    solve_problem_3()
    print("\nTodos os exercícios foram processados e os resultados foram salvos em './result'.")
```