Universidade Federal do Ceará - Sobral

Curso de Engenharia de Computação - Inteligência Computacional 2023.2

Robson Mesquita Gomes - 399682

Sistema de Frenagem com Lógica Fuzzy

Introdução

O projeto se resume em projetar e construir um sistema básico de frenagem utilizando a lógica fuzzy. A programação do sistema foi feita com a ferramenta MATLAB R2023b e recomenda-se que os testes com o mesmo código sejam realizados em uma versão igual ou superior a essa.

Fundamentação Teórica

A lógica difusa, também chamada de lógica fuzzy, é uma forma de lógica multivalorada em que os valores verdade das variáveis podem ser números reais entre 0 (falso) e 1 (verdadeiro), ao contrário da lógica booleana, que usa apenas 0 ou 1. Ela imita o raciocínio humano ao lidar com a verdade parcial, situada entre o totalmente verdadeiro e o totalmente falso. Foi proposta por Lotfi A. Zadeh em 1965 e é usada em controle de processos industriais e inteligência artificial, permitindo lidar com conceitos não quantificáveis, como a temperatura de uma caldeira ou o sentimento de felicidade. Alguns pesquisadores questionam se a lógica difusa é uma verdadeira lógica, devido às soluções aproximadas que pode gerar.

Metodologia

A lógica de execução do projeto pode ser observada a partir da função main (Figura 1).

Figura 1: Função main

```
function main()

    desenho_carro = "   _____\n  /|_||_\\`.__\n (   _    _ _\\\n=.-'|_|--|_|-'";

    % Apresentação
    fprintf("\n\n-- Sistema de Frenagem com Lógica Fuzzy --\n\n");
    fprintf(desenho_carro + "\n\n");

    % Entrada de dados
    pedal   = input("\nQual o valor da pressão no pedal? (de 0 a 100) \n= ");
    roda    = input("\nQual o valor da velocidade da roda? (de 0 a 100) \n= ");
    carro   = input("\nQual o valor da velocidade do carro? (de 0 a 100) \n= ");

    % Pertinências
    [pressao_high,  pressao_med,    pressao_low ]    = pertinencia_pedal(pedal);
    [roda_fast,     roda_med,       roda_low    ]    = pertinencia_roda(roda);
    [carro_fast,    carro_med,      carro_low   ]    = pertinencia_carro(carro);

    % Regras fuzzy

    % SE (pressão média),
    % ENTÃO aplicar o freio
    aperte_1 = pressao_med;

    % SE (pressão alta
    % E velocidade do carro for alta
    % E velocidade das rodas for alta),
    % ENTÃO aplicar o freio
    aperte_2 = min([pressao_high, roda_fast, carro_fast]); % O AND é definido como o mínimo

    % SE (pressão alta
    % E velocidade do carro for alta
    % E a velocidade das rodas for baixa),
    % ENTÃO liberar o freio
    libera_1 = min([pressao_high, roda_low, carro_fast]); % O AND é definido como o mínimo

    % SE (pressão no pedal for baixa),
    % ENTÃO liberar o freio
    libera_2 = pressao_low;

    % Intervalo
    intervalo = 0:1:100;

    % Pertinências liberação
    aperte = aperte_1 + aperte_2;
    libere = libera_1 + libera_2;

    % Pressão no freio
    freio = calcular_pressao_freio(intervalo, aperte, libere);

    % Pressão a ser aplicada no freio (centroid)
    frenagem = sum(freio .* intervalo) / sum(freio);

    % Saída
    fprintf('\nA pressão aplicada no freio é: %.2f;\n\n\n', frenagem);

end
```

Fonte: produção do autor

Pertinência

As pertinências de cada instância recebida é definida por funções específicas utilizadas na função main.

Pertinência da pressão no pedal

A pertinência da pressão no pedal é dada pela função pertinencia_pedal (Figura 2).

Figura 2: Função pertinencia_pedal

```
65    % Calcula pertinência da pressão no pedal
66    function [pressao_pedal_high, pressao_pedal_med, pressao_pedal_low] = pertinencia_pedal(ref_pedal)
67
68        if (ref_pedal > 0 && ref_pedal <= 30)
69            pressao_pedal_low   = 1 - 0.02 * ref_pedal;
70            pressao_pedal_med   = 0;
71            pressao_pedal_high  = 0;
72
73        elseif (ref_pedal > 30 && ref_pedal <= 50)
74            pressao_pedal_low   = 1 - 0.02 * ref_pedal;
75            pressao_pedal_med   = 0.05 * ref_pedal - 1.5;
76            pressao_pedal_high  = 0;
77
78        elseif (ref_pedal > 50 && ref_pedal <= 70)
79            pressao_pedal_low   = 0;
80            pressao_pedal_med   = 3.5 - 0.05 * ref_pedal;
81            pressao_pedal_high  = 0.02 * ref_pedal - 1;
82
83        elseif (ref_pedal > 70 && ref_pedal <= 100)
84            pressao_pedal_low   = 0;
85            pressao_pedal_med   = 0;
86            pressao_pedal_high  = 0.02 * ref_pedal - 1;
87        end
88
89    end
```

Fonte: produção do autor

Pertinência da velocidade da roda

A pertinência da velocidade da roda é dada pela função pertinencia_roda (Figura 3).

Figura 3: Função pertinencia_roda

```
91    % Calcula pertinência da velocidade da roda
92    function [velocidade_roda_fast, velocidade_roda_med, velocidade_roda_slow] = pertinencia_roda(ref_roda)
93
94        if (ref_roda > 0 && ref_roda <= 20)
95            velocidade_roda_slow    = 1 - (1 / 60) * ref_roda;
96            velocidade_roda_med     = 0;
97            velocidade_roda_fast    = 0;
98
99        elseif (ref_roda > 20 && ref_roda <= 40)
100           velocidade_roda_slow    = 1 - (1 / 60) * ref_roda;
101           velocidade_roda_med     = (1 / 30) * (ref_roda - 20);
102           velocidade_roda_fast    = 0;
103
104       elseif (ref_roda > 40 && ref_roda <= 50)
105           velocidade_roda_slow    = 1 - (1 / 60) * ref_roda;
106           velocidade_roda_med     = (1 / 30) * (ref_roda - 20);
107           velocidade_roda_fast    = (1 / 60) * (ref_roda - 40);
108
109       elseif (ref_roda > 50 && ref_roda <= 60)
110           velocidade_roda_slow    = 1 - (1 / 60) * ref_roda;
111           velocidade_roda_med     = (1 / 30) * (80 - ref_roda);
112           velocidade_roda_fast    = (1 / 60) * (ref_roda - 40);
113
114       elseif (ref_roda > 60 && ref_roda <= 80)
115           velocidade_roda_slow    = 0;
116           velocidade_roda_med     = (1 / 30) * (80 - ref_roda);
117           velocidade_roda_fast    = (1 / 60) * (ref_roda - 40);
118
119       elseif (ref_roda > 80 && ref_roda <= 100)
120           velocidade_roda_slow    = 0;
121           velocidade_roda_med     = 0;
122           velocidade_roda_fast    = (1 / 60) * (ref_roda - 40);
123       end
124
125   end
126
```

Fonte: produção do autor

Pertinência da velocidade do carro

A pertinência da velocidade do carro é dada pela função pertinencia_carro (Figura 4).

Figura 4: Função pertinencia_carro

```
127     % Calcula pertinência da velocidade do carro
128     function [velocidade_carro_fast, velocidade_carro_med, velocidade_carro_slow] = pertinencia_carro(ref_carro)
129
130         if (ref_carro > 0 && ref_carro <= 20)
131             velocidade_carro_slow   = 1 - (1 / 60) * ref_carro;
132             velocidade_carro_med    = 0;
133             velocidade_carro_fast   = 0;
134
135         elseif (ref_carro > 20 && ref_carro <= 40)
136             velocidade_carro_slow   = 1 - (1 / 60) * ref_carro;
137             velocidade_carro_med    = (1 / 30) * (ref_carro - 20);
138             velocidade_carro_fast   = 0;
139
140         elseif (ref_carro > 40 && ref_carro <= 50)
141             velocidade_carro_slow   = 1 - (1 / 60) * ref_carro;
142             velocidade_carro_med    = (1 / 30) * (ref_carro - 20);
143             velocidade_carro_fast   = (1 / 60) * (ref_carro - 40);
144
145         elseif (ref_carro > 50 && ref_carro <= 60)
146             velocidade_carro_slow   = 1 - (1 / 60) * ref_carro;
147             velocidade_carro_med    = (1 / 30) * (80 - ref_carro);
148             velocidade_carro_fast   = (1 / 60) * (ref_carro - 40);
149
150         elseif (ref_carro > 60 && ref_carro <= 80)
151             velocidade_carro_slow   = 0;
152             velocidade_carro_med    = (1 / 30) * (80 - ref_carro);
153             velocidade_carro_fast   = (1 / 60) * (ref_carro - 40);
154
155         elseif (ref_carro > 80 && ref_carro <= 100)
156             velocidade_carro_slow   = 0;
157             velocidade_carro_med    = 0;
158             velocidade_carro_fast   = (1 / 60) * (ref_carro - 40);
159         end
160
```

Fonte: produção do autor

Valor de saída

O valor de saída do sistema é definido na função calcular_pressao_freio (Figura 5).

Figura 5: Função calcular_pressao_freio

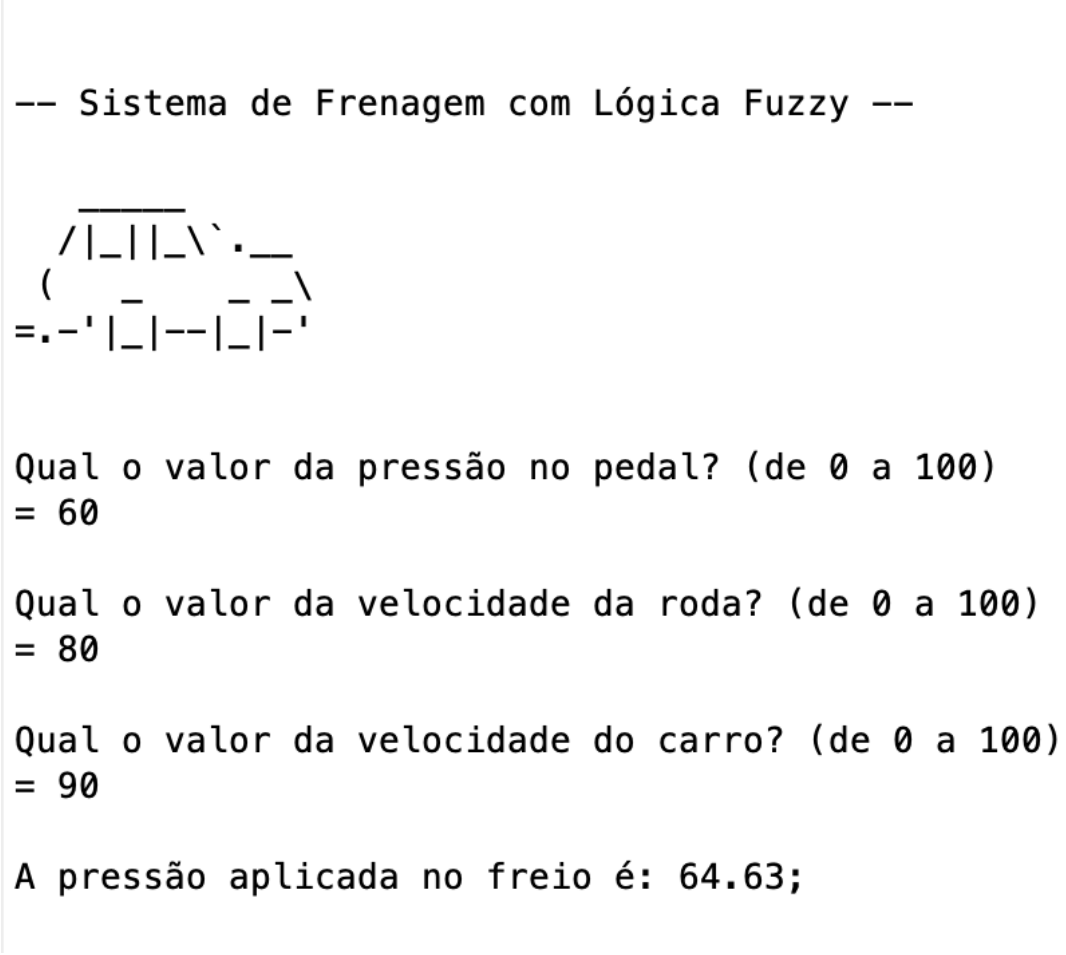
```
162
163     % Retorna a pressão no freio
164     function freio = calcular_pressao_freio(intervalo, aperte_freio, libere_freio)
165
166         freio   = zeros(1, length(intervalo));
167         aperte     = 0.01 * intervalo;
168         libere     = 1 - 0.01 * intervalo;
169
170         for i = 1:length(intervalo)
171             aperte(i)   = min([aperte(i), aperte_freio]);
172             libere(i)   = min([libere(i), libere_freio]);
173             freio(i)    = max([aperte(i), libere(i)]);
174         end
175
176     end
```

Fonte: produção do autor

Conclusão e resultados

O resultado de uma execução válida do projeto pode ser verificado na Figura 6.

Figura 6: Execução do sistema de frenagem



Fonte: produção do autor

O que pode ser observado é que a lógica Fuzzy possui aplicação viável e eficiente para problemas práticos de tomada de decisão.

Bibliografia

Stuart Russel e Peter Norvig, **Inteligência Artificial**, 3ª edição, Editora Campus, 2013.