AVALIAÇÃO DA APLICAÇÃO DE TÉCNICAS DE LÓGICA *FUZZY* NO CONTROLO DE MÁQUINAS ELÉCTRICAS

Jaime Fonseca João Afonso Júlio Martins Carlos Couto

Departamento de Electrónica Industrial

Universidade do Minho

4700 - Braga

Portugal

RESUMO

Este artigo descreve o trabalho desenvolvido na avaliação da aplicação de técnicas de lógica *fuzzy* ao controlo de máquinas eléctricas, e mais concretamente a um motor de indução trifásico. É feita uma apresentação do Matlab/Simulink usado para projectar e simular todo o sistema baseado na técnica referida. É igualmente apresentado a ferramenta informática *fuzzy*TECH que foi usada para projectar, afinar e posteriormente gerar o código correspondente ao sistema de controlo *fuzzy* para o microcontrolador 80C196KC da Intel, que posteriormente será utilizado no *hardware* onde será efectuada a respectiva implementação. É ainda comparado o desempenho do controlador *fuzzy* com um controlador PI.

Palavras chave : Matlab/Simulink, *fuzzy*TECH, Lógica *Fuzzy*.

1. Introdução

Nos últimos anos a lógica *fuzzy* (difusa) começou a surgir como uma técnica alternativa no controlo de processos industriais complexos, e nos mais diversos equipamentos electrónicos e electrodomésticos. A lógica *fuzzy* é um superconjunto da lógica booleana convencional que foi extendida para gerir o conceito de “parcialmente verdade” (valores entre o "completamente verdade" e o "completamente falso"). Foi introduzida pelo Dr. Lofti Zadeh da UC/Berkeley, em 1960, como meio de modelar o conhecimento subjectivo, o qual representa informações linguísticas que normalmente são impossíveis de quantificar usando os métodos mateméticos tradicionais. As aproximações utilizando lógica *fuzzy* permitem ao projectista gerir eficientemente conhecimentos objectivos e subjectivos (dados numéricos e conhecimentos expressos a partir de expressões linguísticas), aplicando-os aos problemas de controlo em malha fechada, reduzindo desta forma o tempo de projecto e os seus custos.

Até agora, em automação industrial a lógica *fuzzy* foi principalmente usada em processos relativamente lentos. Este artigo descreve o trabalho desenvolvido na tentativa de aplicar estas técnicas a processos rápidos, nomeadamente ao controlo de máquinas eléctricas, e mais especificamente do motor de indução trifásico, sem a utilização de hardware específico, nomedamente controladores *fuzzy*. É ainda dado

especial realce às ferramentas informáticas de desenvolvimento utilizadas nomedamente, o Matlab e o *fuzzy*TECH.

2. Ferramentas de simulação e desenvolvimento

A simulação dos sistemas físicos é de grande importância nos campos teóricos e/ou aplicados. A simulação possibilita a cientistas e engenheiros, em momentos, analisar o comportamento dos sistemas físicos a partir do correspondente diagrama de blocos ou do modelo matemático, permitindo desta forma uma diminuição dos tempos de desenvolvimento com uma consequente diminuição dos custos.
As ferramentas de simulação e desenvolvimento tem um papel muito importante quando se projectam sistemas de controlo, assumindo ainda uma importância mais relevante quando se projectam controladores *fuzzy*, pois estes são projectados a partir de conhecimentos subjectivos fornecidos por peritos e por conseguinte o seu afinamento é totalmente dependente dos meios de simulação.
O Matlab é um ambiente que combina computação numérica, visualização gráfica e linguagens de alto nível. É um ambiente natural para análise, prototipagem de algoritmos e desenvolvimento de aplicações. O Matlab dispõe de um conjunto de bibliotecas que disponibilizam funções para resolver problemas particulares, denominadas *Toolboxes*.
Neste caso foram utilizadas as seguinte *tooloboxes* do Matlab: *Nonlinear Control Design Toolbox* (para afinar o controlador PI com mecanismo de *reset windup*) e a *Identification Toolbox* (para obtenção do modelo do conjunto inversor-motor de indução trifásico).
O Simulink está construído sobre o Matlab, sendo um ambiente interactivo para modelação, análise e simulação de uma grande variedade de sistemas dinâmicos.

O *fuzzyTECH MCU-96 edition*, usado para desenvolver o controlador *fuzzy*, cobre todas as fases do projecto de um controlador deste tipo. Esta ferramenta permite definir o projecto, as variáveis linguísticas, as regras, e efectuar uma simulação interactiva. Em seguida, esta ferramenta permite gerar código ANSI C, Kernighan and Richie C e Assembler neste caso para o microcontrolador INTEL da família 96. Versões do fuzzyTECH para outros microcontroladores também se encontram disponíveis.

3. Controlo de deslizamento no motor de indução

A figura 1 apresenta um esquema convencional de um controlador de deslizamento para o motor de indução, o qual é usado em variadores de velocidade sem grande exigência de desempenho. Tradicionalmente, o erro de velocidade (ω_e) é a entrada do controlador PI, que estabelece a frequência de deslizamento do motor (ω_r).
A frequência estatórica resulta da adição da frequência de deslizamento à velocidade do rótor (ω). A tensão do estator (U_s) é estabelecida de acordo com uma lei pré-definida (U_s/ω_s aproximadamente constante), de forma que o fluxo do motor seja mantido no seu valor nominal. A frequência de deslizamento é limitada, estabelecendo (indirectamente) um limite para o binário e para a corrente do estator.

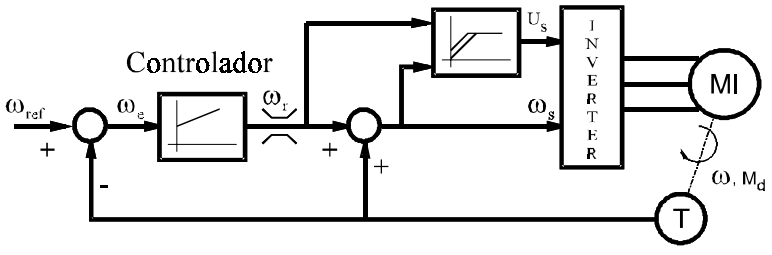


Figura 1. - Esquema convencional de um controlador de deslizamento

A ideia apresentada foi simulada em ambiente Matlab, tendo sido utilizadas técnicas *fuzzy* para implementar o controlador. Os resultados obtidos são comparados com os de um controlador convencional (PI com mecanismo de *reset windup*).
A figura 2 apresenta o sistema de diagrama de blocos usado no ambiente Simulink. Esta ferramenta disponibiliza uma biblioteca com diferentes blocos, diversos algoritmos de integração e permite ao utilizador uma fácil selecção dos parâmetros de simulação.

O bloco "Controlador de Deslizamento" contêm toda a computação numérica e o controlador *fuzzy* necessários à implementação do controlador de deslizamento como é mostrado na figura 3. O bloco "Motor+Carga" contém o modelo do motor de indução e da carga determinados através da *Identification Toolbox*.

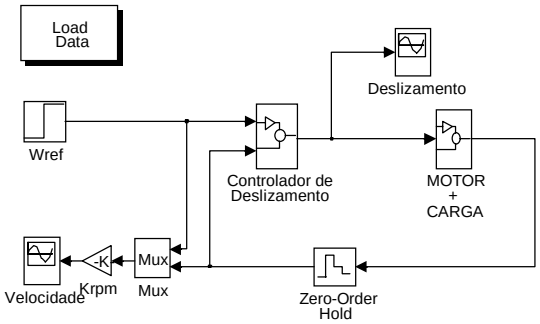


Figura 2. - Diagrama de blocos usado no Simulink

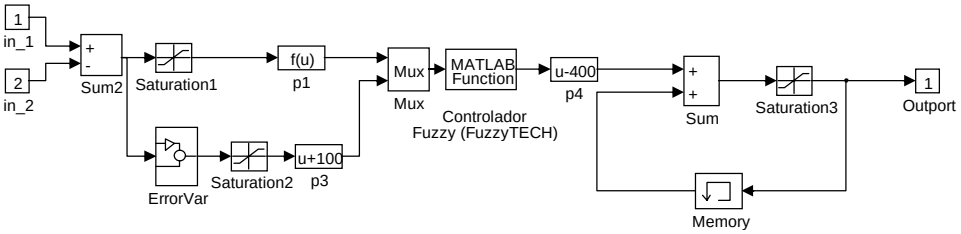


Figura 3. - Diagrama de blocos que constitui o controlador de deslizamento

O controlador *fuzzy* foi implementado no *fuzzy*TECH, que gera código M (caracteristico do Matlab), que permite tratá-lo no Simulink como se fosse uma função M. Foi utilizado o *fuzzy*TECH em detrimento da *toolbox* de lógica *fuzzy* do Matlab devido aos seguintes motivos:

- é o *fuzzy*TECH que permite gerar o código optimizado para o microcontrolador 80C196 da Intel e pretende-se que a simulação seja o mais próxima possível da situação real;
- a *toolbox* de *fuzzy* do Matlab não tem implementado o método de desfuzificação que foi utilizado e que possibilita a obtenção de um código mais optimizado (CoM - Center of Maximum).

O diagrama de blocos utilizado na simulação do controlador PI é idêntico ao mostrado na figura 2, somente o bloco do controlador de deslizamento é trocado por um bloco que implementa o controlador PI. É de salientar que o PI foi afinado utilizando uma outra *toolbox* do Matlab denominada *Nonlinear Control Design*.

4. Desenvolvimento do controlador *fuzzy* com o *fuzzy*TECH

Como já anteriormente referido o *fuzzyTECH MCU-96 Edition* dispõe de todas as fases necessárias ao desenvolvimento de um controlador *fuzzy*.

A. Definição do projecto

O primeiro passo quando se usa o *fuzzyTECH MCU-96 Edition* é definir a estrutura do controlador por meio da janela *Project Editor*.
A Figura 4 mostra a estrutura do controlador e permite ao projectista ter acesso directo à definição das variáveis linguísticas e das regras.

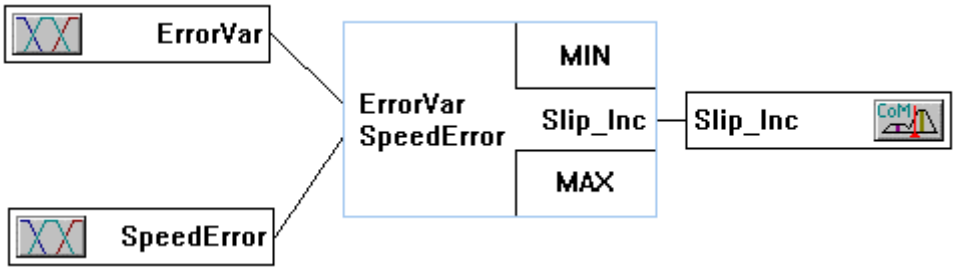
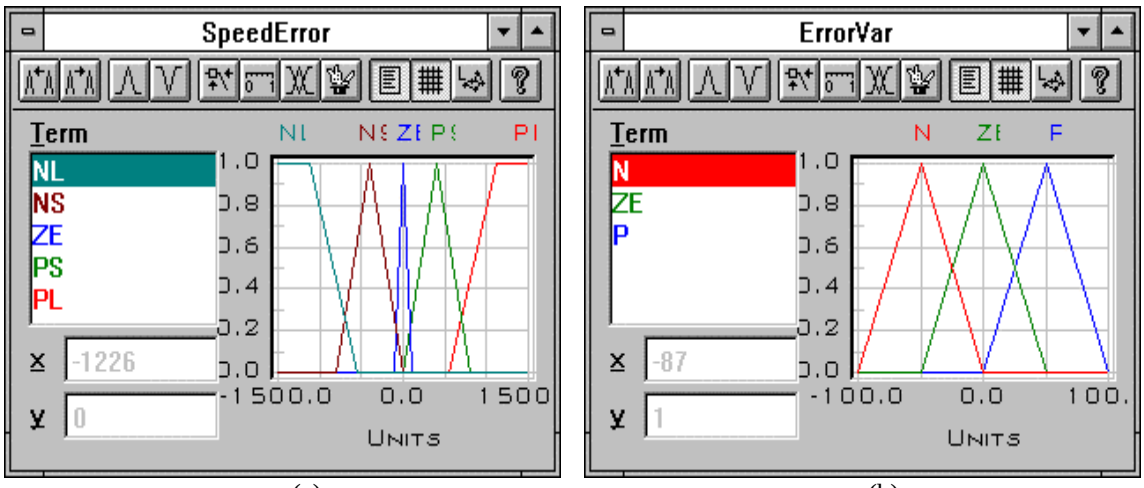


Figura 4. - Estrutura do controlador de deslizamento para o motor de indução

B. Definição das variáveis linguísticas

O próximo passo é a definição das variáveis linguísticas. O interface gráfico do *fuzzy*TECH permite ao projectista facilmente criar as variáveis linguísticas mais comuns e as funções pertença para a aplicação. O controlador de deslizamento, como mostrado na Figura 4, tem duas entradas, o Erro de Velocidade (*SpeedError*) e a Variação do Erro (*ErrorVar*), e uma saída, que é o Incremento do Deslizamento (*Slip_Inc*).



(a) (b)

Figura 5. - Funções pertença para o Erro de Velocidade e para a Variação do Erro

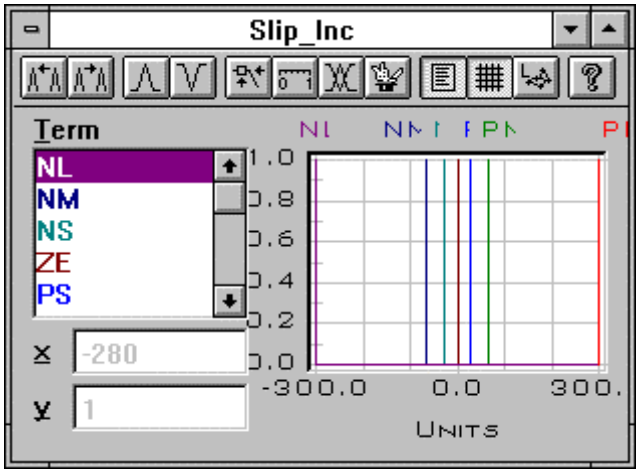


Figura 6. - Funções pertença para o Incremento do Deslizamento

O Erro de Velocidade é descrito por 5 funções pertença triangulares: NL (*Negative Large*), NS (*Negative Small*), ZE (*Zero*), PS (*Positive Small*) e PL (*Positive Large*). A Variação do Erro é descrita por 3 funções pertença triangulares: N (*Negative*), ZE (*Zero*) e P (*Positive*). A Incrementação do Deslizamento é

descrita por 7 funções pertença triangulares: NL (*Negative Large*), NM (*Negative Medium*), NS (*Negative Small*), ZE (*Zero*), PS (*Positive Small*), PM (*Positive Medium*) e PL (*Positive Large*). As funções pertença de entrada mostradas na Figura 5 são definidas tendo em conta a velocidade, a aceleração do motor e a resolução do sistema.

Durante a definição das variáveis linguísticas, o *fuzzy*TECH permite ao utilizador definir duas representações para as variáveis: *shell values* e *code values*. Os primeiros são os valores do mundo real que as variáveis representam, sendo somente usados para mostrar os valores actuais com o *fuzzy*TECH. Os segundos são os valores internos de 16 bits que o microcontrolador usa para calcular os resultados, e a sua gama pode ir de 0 a 65535. Se a escala para os *code values* e para os *shell values* for a mesma é mais fácil perceber o comportamento do controlo por comparação directa das entradas/saídas *fuzzy* reais com as variáveis linguísticas.

C. Definição das regras

A Figura 7 mostra o Editor de Regras com as regras definidas para o controlador de deslizamento. A definição das regras permite compreender melhor o comportamento do sistema. Foram estabelecidas regras para manter o erro de velocidade (*SpeedError*) perto de zero, regras para evitar o *overshoot* na velocidade do motor e regras que fornecem uma rápida resposta quando o erro de velocidade é grande.

Spreadsheet Rule Editor - RB1

	IF		THEN	
	ErrorVar	SpeedError	DoS	Slip_Inc
1		PL	1.00	PL
2	N	NS	1.00	NM
3	N	ZE	1.00	NS
4	N	PS	1.00	NM
5	ZE	NS	1.00	NS
6	ZE	ZE	1.00	ZE
7	ZE	PS	1.00	PS
8	P	NS	1.00	PM
9	P	ZE	1.00	PS
10	P	PS	1.00	PM
11		NL	1.00	NL

Figura 7 - Editor de regras

D. Optimização do comportamento do sistema

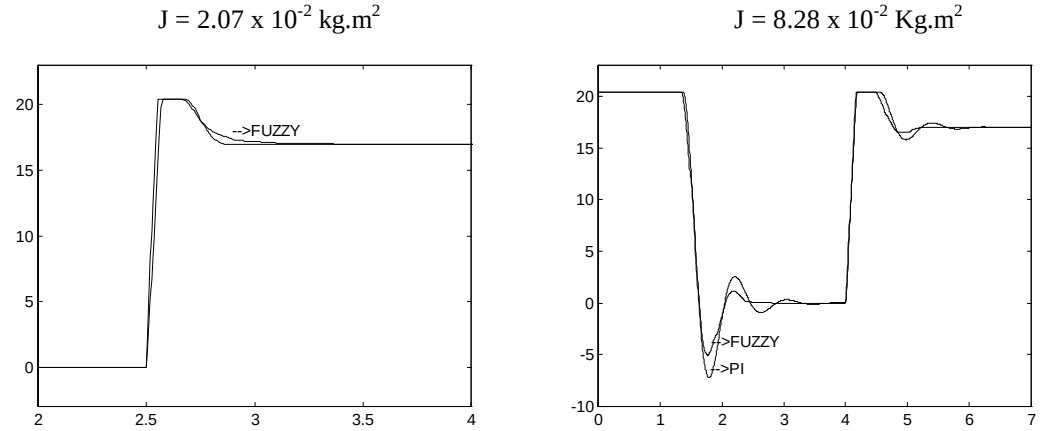
O *fuzzy*TECH fornece mecanismos *off-line* para *debug*, teste e optimização das regras e funções pertença. O modo de *debug* interactivo oferece uma verificação gráfica de todos os passos do projecto. O modo *batch* permite criar um ficheiro de saída associado a um ficheiro de entradas que contém uma amostra dos valores possíveis que as entradas podem ter. Este modo permite avaliar e testar o desempenho do projecto.
O *fuzzy*TECH fornece ainda um mecanismo de *debug on-line* através de comunicação RS-232.
Usando estas características, o projectista pode verificar se todas as regras definidas são necessárias, se regras importantes não foram esquecidas e se variações das variáveis de saída são consistentes com o sistema.

O código para o microcontrolador 80C196 da Intel a usar é automaticamente gerado pelo *fuzzy*TECH.

5. Resultados da simulação

Em seguida são apresentados e comparados os resultados da simulação de um controlador *fuzzy* e de um controlador convencional (PI com mecanismo de *reset windup*), sintonizado com a *Nonlinear System Design Toolbox*. Resposta a um degrau de carga de 0% para 100% do binário nominal.

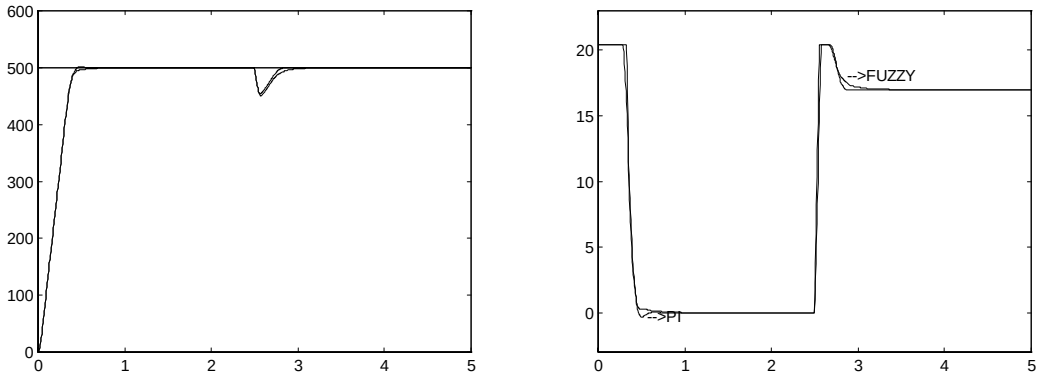
Todas as simulações são efectuadas com um período de amostragem igual a 5 ms



(a) (b)

Figura 8 - Resposta a um degrau de binário: (a) J = 2.07 x 10^{-2} kg.m^2; (b) J = 8.28 x 10^{-2} Kg.m^2

A figura 9 apresenta a velocidade do motor e o deslizamento durante o arranque com 10% do binário nominal e J = 2.07 x 10^{-4} kg.m^2 quando é usado respectivamente os controladores *fuzzy* e PI. Neste caso a resposta de ambos os controladores é semelhante.



(a) (b)

Figura 9 - Arranque do motor: (a) velocidade; (b) Deslizamento

6. Conclusões

Neste artigo foi feita uma avaliação das técnicas de lógica *fuzzy* aplicadas ao controlo de um motor de indução trifásico. O Matlab/Simulink foi usado para simular todo o sistema. O *fuzzy*Tech foi usado para

projectar o controlador *fuzzy* e posteriormente será usado para gerar o código para o microcontrolador 80C196KC da Intel.
Os resultados da simulação confirmam uma performance dinâmica superior do controlador com lógica *fuzzy* comparativamente com o esquema mais convencional utilizado (controlador PI), nomeadamente em termos de insensibilidade a alterações dos parâmetros do modelo.

O período de amostragem usado com o controlador de deslizamento - 5ms - provavelmente será compatível com o controlo em tempo real usando *hardware standard*.

Referências

[1] Jerry M. Mendel, "Fuzzy Logic Systems for Engineering: A Tutorial", Proceedings of the IEEE, Vol. 83, No. 3, pp 345-377, 1995.

[2] David Brubaker, "Design and Simulate your Own Fuzzy Setpoint Controller", EDN, pp 167-170, January 5, 1995.

[3] David Brubaker, "Simulation in Fuzzy-Controller Design", EDN, pp 163-166, February 16, 1995.

[4] David Brubaker, "Fuzzy Setpoint Controller-Act 3", EDN, pp 133-136, March 16, 1995.

[5] H. Chris Tseng and Victor H. Hwang, "Servocontroller Tunning with Fuzzy Logic", IEEE Transaction on Control Systems Technology, Vol. 1, No. 4, pp 262-269, 1993.

[6] Howard Demuth and Mark Beale, "Neural Network Toolbox For Use with MATLAB", The Math Works, Inc., 1993.

[7] J.-S. Roger Jang and Ned Gulley, "Fuzzy Logic Toolbox For Use with MATLAB", The Math Works, Inc., 1995.

[8] J. S. Martins, "Controlo de Velocidade do Motor de Indução Trifásico", PhD Thesis, Minho University (Portugal), 1993.

[9] Pierre Guillemin, "Fuzzy Logic Applied to Motor Control", IEEE Transaction on Industry Applications, Vol. 32, No. 1, pp 51-56, January/February 1996.

[10] Jaime Fonseca, João L. Afonso, Júlio S. Martins, Carlos Couto, "Evaluation Of Neural Networks and Fuzzy Logic Techniques Applied to the Control Of Electrical Machines", Proceedings of the Mechatronics'96, July 1996.