Proceedings of the VI Brazilian Conference on Neural Networks - VI Congresso Brasileiro de Redes Neurais
pp. 259–264, June 2–5, 2003 – Centro Universitário da FEI, São Paulo - SP - Brazil

Controlador Preditivo Baseado em Otimização por Colônia de Partículas

Leandro dos Santos Coelho[1], Renato A. Krohling[2]
[1] Pontifícia Universidade Católica do Paraná
Grupo Produtrônica, Programa de Pós-Graduação em Engenharia de Produção e Sistemas
Laboratório de Automação e Sistemas, LAS/PPGEPS/CCET/PUCPR
Rua Imaculada Conceição, 1155, CEP 80215-901, Curitiba, PR, Brasil
[2] Universidade Federal do Espírito Santo, Departamento de Engenharia Elétrica
Campus de Goiabeiras, C.P. 01-9001, CEP 29060-970, Vitória, ES, Brasil
E-mails: lscoelho@rla01.pucpr.br, renato@ele.ufes.br

Abstract

Generalized predictive controllers (GPCs) have been successfully applied in complex processes control during the last decade. This paper presents the design of an adaptive GPC using any new approaches of particle swarm optimization algorithm based on Gaussian and Cauchy distributions. Steps of the optimization procedure, control law implementation and adaptive procedure are discussed. Simulation results for the control of a nonlinear process of Hammerstein type shown the efficiency of proposed optimization methodology.

1. Introdução

Nos últimos anos algumas pesquisas têm sido relatadas quanto ao desenvolvimento de métodos de projeto e otimização de controladores preditivos baseados em modelo (*MBPCs*) usando algoritmos genéticos [1]-[7]. Neste artigo, os aspectos de projeto, otimização e desempenho de um *MBPC* — o controlador preditivo generalizado (*GPC)* adaptativo [8], [9] — usando uma abordagem rápida de otimização por colônia de partículas (*particle swarm optimization*) [10]-[12] são propostos e discutidos.

A otimização por colônia de partículas (*OCP*) é uma abordagem de algoritmo evolutivo baseada em uma população de indivíduos e motivada pela simulação de comportamento social em vez da sobrevivência do indivíduo mais apto. Na *OCP*, cada solução candidata (denominada partícula) possui associada uma velocidade. A velocidade é ajustada através de uma equação de atualização que considera a experiência da partícula correspondente e a experiência das outras partículas da população.

O estudo de caso de otimização do *GPC* usando diversas abordagens de *OCP* é um processo não-linear que possui características não-lineares do tipo Hammerstein. Nas próximas seções o projeto do *GPC* adaptativo e a descrição da *OCP* são apresentados. O projeto e os resultados de simulação para o comportamento servo do sistema de controle, em malha fechada, são também discutidos.

2. Controlador preditivo

Os *MPBC*s apresentam uma série de vantagens sobre outros métodos de projeto de controle. O *MPBC* é particularmente atraente no tratamento de processos complexos, com características de atraso de transporte, fase não-mínima, instabilidade em malha aberta e apresentando restrições e incertezas [1], [8].

As metodologias de *MBPC* utilizam o modelo de processo incorporando o comportamento futuro previsto do processo no procedimento de projeto do controlador. Diversas metodologias de *MBPC* têm sido estudadas, mas as técnicas possuem em comum a estrutura básica de projeto. As principais diferenças são relativas à forma em que as previsões futuras são calculadas, o tipo de modelo matemático utilizado e como a função custo é definida. A seguir é mencionado o projeto de um *GPC* na concepção adaptativa com o algoritmo clássico dos mínimos quadrados recursivo para identificação do modelo matemático do processo.

2.1. Equacionamento e projeto do *GPC*

O *GPC* constitui-se de uma generalização dos algoritmos preditivos de múltiplos passos (*multi-step-ahead*). Além disso, o *GPC* é também uma extensão natural com horizonte estendido do controle *single-step-ahead*, denominado controlador de variância mínima generalizada proposto por Clarke & Gawthrop [13]. O *GPC* realiza a previsão da saída do processo em um horizonte de tempo maior que o atraso de transporte do processo. O sinal de controle é calculado pela minimização de uma função custo (quadrática) que é obtida da diferença entre a saída prevista e o sinal de referência, com restrições sobre as ações de controle futuras.

A maioria das estratégias de controle preditivo é basicamente uma combinação de: (i) um modelo, que descreve o processo, necessário a previsão das saídas futuras do processo; (ii) uma função custo, usualmente,

quadrática, considerando o controle e os erros futuros da saída do processo. Esta função é calculada para um número fixo de passos a frente para prever o comportamento do processo; (iii) uma trajetória conhecida das saídas de referências futuras; (iv) restrições do sistema e variáveis de controle (opcional); e (v) um algoritmo de otimização para a escolha da seqüência de controle que minimiza a função custo.

No presente estudo, considera-se o *GPC* com a formulação incremental de maneira a evitar o erro em regime, muitas vezes, presente na formulação posicional. Os controladores preditivos *multi-step-ahead* utilizam um modelo para o processo do tipo *CARIMA* (*Controlled Auto-Regressive Integrated Moving Average*) conforme representado pela seguinte equação a diferenças [14]:

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + C(q^{-1})\xi(t)/\Delta \quad (1)$$

que pode ser reescrita na forma,

$$A(q^{-1})\Delta y(t) = q^{-d}B(q^{-1})\Delta u(t) + C(q^{-1})\xi(t) \quad (2)$$

onde

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a},$$
$$B(q^{-1}) = b_0 + b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b},$$
$$C(q^{-1}) = c_0 + c_1 q^{-1} + \ldots + c_{n_c} q^{-n_c},$$

e $y(t)$ é a saída do sistema, $u(t)$ é o sinal de controle, $\xi(t)$ é uma seqüência aleatória do tipo ruído branco e $\Delta=(1 - q^{-1})$. O projeto adotado para o *GPC* adaptativo é baseado na abordagem proposta por Clarke *et al.* [8], [9]. Neste caso, a função custo é dada pela equação:

$$J_{GPC}=E\left\{\sum_{j=N_1}^{N_2}[y(t+j)-y_r(t+j)]^2+\sum_{j=1}^{N_u}\Gamma\Delta u^2(t+j-1)\right\}, \quad (3)$$

onde Γ é a ponderação do sinal de controle, N_1 e N_2 são os horizontes de previsão da saída inicial e final, respectivamente, e N_u é o horizonte de controle. Os termos $y(t+j)$ e $y_r(t+j)$ representam o sinal da saída e o sinal de referência j passos a frente e, $\Delta u(t+j-1)$ é o incremento do sinal de controle no instante $(t+j-1)$.

Os horizontes de previsão e a ponderação do controle são os principais parâmetros de sintonia do *GPC*. A partir da seleção destes parâmetros é possível obter-se diferentes tipos de controladores preditivos e ajustar o desempenho desejado para o sistema controlado. Usando-se as seguintes equações polinomiais

$$1 = F_j\overline{A} + z^{-j}G_j \text{ e} \quad (4)$$

$$BF_j = E_j + z^{-j}H_j, \quad (5)$$

onde $j = 1, 2,..., N_2$, é possível escrever-se

$$y(t+j) = E_j\Delta u(t+j-1) + G_j y(t) + H_j\Delta u(t-1). \quad (6)$$

Na forma vetorial a equação (6) é dada por

$$\boldsymbol{y} = E\boldsymbol{u} + Gy(t) + H\Delta u(t-1) \quad (7)$$
$$\boldsymbol{y}^T = [y(t+1),..., y(t+N_2)] \quad (8)$$
$$\boldsymbol{u}^T = [\Delta u(t),..., \Delta u(t+N_2-1)] \quad (9)$$
$$G^T = [G_1,..., G_{N2}] \quad H^T = [H_1,..., H_{N2}] \quad (10)$$

e sendo E uma matriz triangular inferior (N_2xN_2) onde

$$E = \begin{bmatrix} e_0 & & & \\ e_1 & e_0 & & \\ \ldots & & & \\ e_{N2-1} & e_{N2-2} & \ldots & e_0 \end{bmatrix} \quad (11)$$

A partir das definições de $\boldsymbol{y}$, $\boldsymbol{u}$ e usando-se

$$\boldsymbol{y_r}^T = [y_r(t+1),..., y_r(t+N_2)] \quad (12)$$

e a função custo é dada por

$$J = \{(\boldsymbol{y}-\boldsymbol{y_r})^T(\boldsymbol{y}-\boldsymbol{y_r}) + \Gamma\boldsymbol{u}^T\boldsymbol{u}\}. \quad (13)$$

Substituindo-se a equação (7) na equação (13) e sendo J minimizada em relação a $\boldsymbol{u}$, resulta na lei de controle:

$$\boldsymbol{u} = (E^TE + \Gamma I)^{-1}E^T[\boldsymbol{y_r} - Gy(t) - H\Delta u(t-1)] \quad (14)$$

Visando-se evitar a singularidade de $(E^TE + \Gamma I)$ e para reduzir-se o esforço computacional para um horizonte de saída com valor elevado, a técnica de impor um horizonte de controle N_u ($N_u < N_2$) é utilizada, ou seja, quando $j \geq N_u$, o controle incremental é $\Delta u(t+j) = 0$ (um controle constante depois do instante de tempo N_u). Embora, o vetor de controle, $\boldsymbol{u}$, e a matriz E (N_2xN_u) tenham a forma:

$$\boldsymbol{u}^T = [\Delta u(t),..., \Delta u(t+N_u-1)] \quad (15)$$

$$E = \begin{bmatrix} e_0 & & & \\ e_1 & e_0 & & \\ \ldots & & & \\ e_{Nu-1} & e_{Nu-2} & \ldots & e_0 \\ \ldots & & & \\ e_{N2-1} & e_{N2-2} & \ldots & e_{N2-Nu} \end{bmatrix} \quad (16)$$

2.2. Abordagem adaptativa do *GPC*

Os algoritmos *GPC* adaptativos, ou mesmo *MBPCs* de forma geral, podem ser obtidos segundo dois procedimentos de projeto: direto e indireto. O caso direto utiliza um estimador recursivo para obter diretamente os parâmetros da lei de controle com uma realimentação a partir das medidas de entrada e saída do processo. O caso indireto combina um estimador recursivo para obter os parâmetros do modelo do processo e um procedimento de projeto do controlador.

Em síntese, no esquema de *GPC* adaptativo, o estimador de parâmetros deve ser iterativo, onde o modelo do sistema é atualizado, a cada período de amostragem, quando novas medidas estão disponíveis.

Neste artigo foi utilizado o algoritmo dos mínimos quadrados recursivo (*MQR*) [15] que pode ser resumido nos seguintes passos: (i) obter a(s) saída(s) e a(s) entrada(s) do processo a ser identificado; (ii) atualizar o vetor de medidas; (iii) calcular o erro de previsão; (iv) calcular o ganho do estimador; (v) atualizar o vetor de parâmetros estimados; (vi) atualizar a matriz de covariância; e (vii) retornar para o passo (i).

2.3. Otimização do *GPC* usando *OCP*

2.3.1 Otimização por colônia de partículas

A *OCP* é uma metodologia da computação evolutiva desenvolvida originalmente por Kennedy & Eberhart em 1995 [10], [11]. A *OCP* é motivada pela simulação do comportamento social em vez da evolução da natureza como em outros algoritmos evolutivos (algoritmos genéticos, programação evolutiva, estratégias evolutivas e programação genética).

A *OCP* é uma metodologia baseada em população de soluções. De forma similar a outros algoritmos evolutivos, a *OCP* é iniciada com uma população de soluções gerada aleatoriamente.

De forma diferente dos algoritmos evolutivos, cada solução potencial (indivíduo) na *OCP* é também atribuída uma velocidade aleatória. As soluções potenciais denominadas *partículas* são então "movimentadas" pelo espaço de busca do problema.

Cada partícula conserva o conhecimento do seu melhor valor da função de aptidão (*fitness*) denotada por *pbest* (versão *local*). Um outro melhor valor é "seguido" pela versão *global*, *gbest*, da *OCP* e sua localização é obtida de alguma partícula que compõe a população.

O conceito da *OCP* consiste de, a cada passo iterativo (geração), mudar a velocidade (acelerando) de cada partícula em direção as localizações do *pbest* e do *gbest*. A aceleração desta busca é ponderada através de um termo gerado de forma aleatória vinculados estes de forma separada as localizações do *pbest* e do *gbest.* O procedimento para implementação da *OCP* é regido pelas seguintes etapas [16]:

(i) iniciar uma população (matriz) de partículas, com posições e velocidades em um espaço de problema *n* dimensional, de forma aleatória com distribuição uniforme.

(ii) para cada particular, avaliar a função de aptidão.

(iii) comparar a avaliação da função de aptidão da partícula com o *pbest* da partícula. Se o valor corrente é melhor que *pbest*, então o valor de *pbest* passa a ser igual ao valor da função de aptidão da partícula, e a localização do *pbest* passa a ser igual a localização atual no espaço *n* dimensional.

(iv) comparar a avaliação da função de aptidão com o prévio melhor valor de aptidão da população. Se o valor atual é melhor que o *gbest*, atualizar o valor de *gbest* para o índice e valor da partícula atual.

(v) modificar a velocidade e a posição da partícula de acordo com as equações (17) e (18), respectivamente onde [17], [18]

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot ud(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Ud(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad , \tag{17}$$

$$\boldsymbol{x}_i = (\boldsymbol{x}_i + \boldsymbol{v}_i) \ . \tag{18}$$

(iv) ir para a etapa (ii) até que um critério de parada seja encontrado, usualmente uma função de aptidão "suficientemente boa" ou um número máximo de iterações seja atingido.

As notações usadas são: $\boldsymbol{x}_i = [x_{i1}, x_{i2}, ..., x_{in}]^{\mathrm{T}}$ armazena a posição da *i*-ésima partícula, $\boldsymbol{v}_i = [v_{i1}, v_{i2}, ..., v_{in}]^{\mathrm{T}}$ armazena a velocidade da *i*-ésima partícula e $\boldsymbol{p}_i = [p_{i1}, p_{i2}, ..., p_{in}]^{\mathrm{T}}$ representa a posição do melhor valor de aptidão da *i*-ésima partícula.

O índice *g* representa o índice da melhor particular entre todas as partículas do grupo. A variável *w* é a ponderação de inércia, c_1 e c_2 são constantes positivas; *ud*() e *Ud*() são duas funções para geração de números aleatórios com distribuição uniforme no [0,1], respectivamente.

As velocidades das partículas em cada dimensão são limitadas a um valor máximo de velocidade, *Vmax*. O *Vmax* é um parâmetro importante, pois determina a resolução que a região próxima a soluções atuais são procuradas. Se *Vmax* é muito alto, a *OCP* facilita a busca global, enquanto um valor *Vmax* pequeno enfatiza as buscas locais.

A primeira parte na equação (17) é um termo de momento da partícula. A ponderação de inércia *w* representa o grau de momento da partícula. A segunda parte consiste da parte "cognitiva", que representa o "conhecimento" da partícula independentemente. A terceira parte é a parte "social", que representa a colaboração entre as partículas.

As constantes c_1 e c_2 representam a ponderação das partes de "cognição' e "social" que influenciam cada particular em direção as *pbest* e *gbest*. Estes parâmetros são usualmente ajustados por heurísticas de tentativa e erro. O tamanho da população também é selecionada dependendo do problema. Adota-se, neste artigo, $c_1 = c_2 = 2{,}0$; *Vmax* para 20% do intervalo da variável de cada dimensão.; e tamanho de população de 30 partículas.

2.3.2 Novas abordagens de *OCP*

As abordagens de *OCP* rápida propostas e avaliadas, neste artigo, baseiam-se nos estudos de operadores de mutação em algoritmos de programação evolutiva rápida (*fast evolutionary programming*) [19], [20], visam modificar a equação (17) da *OCP* convencional

(tipo 1) para utilizá-la com distribuição Gaussiana ou de Cauchy. A utilização de distribuição de Cauchy em algoritmos evolutivos pode ser útil para escapar-se de mínimos locais, enquanto a distribuição de Gauss (normal) pode prover uma convergência mais rápida em buscas locais [21]. As abordagens, para modificação da equação (17) da *OCP* convencional, propostas e avaliadas são:

Tipo 2: Usa função com distribuição de Cauchy modificada, *cd()*, para geração de números aleatórios no intervalo [0,1] da parte "cognitiva":

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot cd(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Ud(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad (19)$$

Tipo 3: Usa função com distribuição de Cauchy modificada, *Cd()*, para geração de números aleatórios no intervalo [0,1] da parte "social":

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot ud(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Cd(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad (20)$$

Tipo 4: Usa função com distribuição de Cauchy modificada, *cd()* e *Cd()*, para geração de números aleatórios no intervalo [0,1], tanto na parte "cognitiva" quanto na "social":

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot cd(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Cd(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad (21)$$

Tipo 5: Usa função com distribuição de Gauss (normal), *gd()*, para geração de números aleatórios no intervalo [0,1] na parte "cognitiva":

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot gd(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Ud(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad (22)$$

Tipo 6: Usa função com distribuição de Gauss (normal), *Gd()*, para geração de números aleatórios no intervalo [0,1] na parte "social":

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot ud(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Gd(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad (23)$$

Tipo 7: Usa função com distribuição de Gauss (normal), *gd()* e *Gd()*, para geração de números aleatórios no intervalo [0,1], tanto na parte "cognitiva" quanto na parte :

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot gd(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Gd(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad (24)$$

Tipo 8: Usa função com *gd()* na parte "cognitiva" e *Cd()* na parte "social" :

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot gd(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Cd(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad (25)$$

Tipo 9: Usa função com *Gd()* na parte "social" e *cd()* na parte "cognitiva" :

$$\boldsymbol{v_i} = w \cdot \boldsymbol{v_i} + c_1 \cdot cd(\,) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) + c_2 \cdot Gd(\,) \cdot (\boldsymbol{p_g} - \boldsymbol{v_i}) \quad (26)$$

3. Estudo de caso para análise do *GPC*: processo não-linear do tipo Hammerstein

O estudo de caso para análise do desempenho do *GPC* é um modelo matemático não-linear do tipo Hammerstein. O modelo matemático não-linear do processo, do tipo Hammerstein adotado, é apresentado em Katende *et al*. [22]. O modelo de Hammerstein é composto por um bloco não-linear estático em série com um bloco linear dinâmico, conforme apresentado na figura 1.

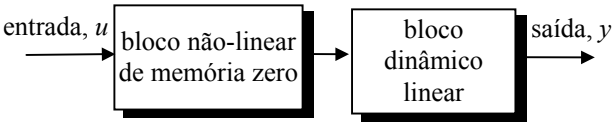

Figura 1: Modelo matemático de Hammerstein.

A dinâmica que rege o comportamento dinâmico do processo estudado neste artigo é dada pela equação a diferenças,

$$\begin{aligned} &(1 + 0{,}022q^{-1} - 0{,}7991q^{-2})y(t) = \\ &q^{-1}(0{,}029 + 0{,}0269)u(t) + \\ &q^{-1}(0{,}021 + 0{,}003q^{-1})u^2(t) \end{aligned} \quad , \quad (27)$$

onde *u(t)* e *y(t)* representam os sinais de entrada e saída do processo, respectivamente.

4. Resultados do *GPC* usando *OCP*

O *GPC* utilizado nas simulações é o com procedimento de projeto indireto, baseado na técnica *self-tuning* (uma abordagem de controle adaptativo).

Neste caso, o controle é calculado supondo-se que os parâmetros do processo são desconhecidos. O estimador dos *MQR* é utilizado para obter os parâmetros do processo a partir das medidas de entrada e saída e, a seguir, substitui-se os parâmetros pelos valores estimados, de forma recursiva. A partir dos parâmetros estimados calcula-se os parâmetros do *GPC*, conforme apresentado na figura 2.

O problema de projetar os melhores valores para os parâmetros N_u, N_2 e Γ é um problema complexo que apresenta objetivos conflitantes para o projeto do *GPC*. Na literatura tem sido apresentados alguns estudos quanto ao projeto e os parâmetros a serem utilizados no *GPC* [23], [24].

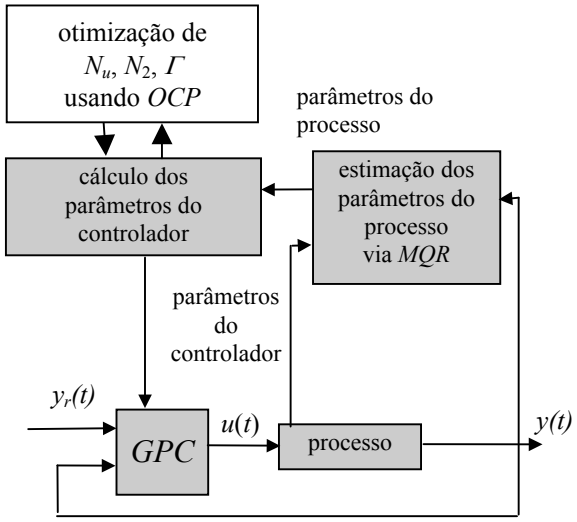

Figura 2: Otimização do *GPC* adaptativo usando *OCP*.

A tabela 1 resume os resultados preliminares obtidos com a otimização de N_u, N_2 e Γ através de 9 configurações de *OCP* mencionadas na seção 2.3.2.

Os intervalos de busca usados foram: $1 \le N_u \le 8$; $1 \le N_2 \le 8$ e $0{,}001 \le \Gamma \le 2{,}000$. Os valores de N_u e N_2 possuem valores reais. Neste caso os valores obtidos pelo *OCP* para estas variáveis são arredondados para valores inteiros. Foram realizados 10 experimentos para cada tipo de *OCP*. O critério de parada foi de 20 gerações (iterações) e uma população de 30 partículas (soluções para o problema) para o estudo de caso abordado. O funcional a ser minimizado pela *OCP* é dado pela equação:

$$J(u,e) = \frac{\sum_{k=1}^{N} i * [e(t)]^2}{N} \qquad (28)$$

onde *e(t)* é o erro entre a saída do processo e o sinal de referência, y_r para a amostra t, N é o número de amostras do processo, w é fixado em 1, i é igual t, exceto quando ocorre alguma mudança de referência, pois neste caso, i=1 e após passa a ser incrementado sucessivamente a cada iteração da mesma forma que t. Conseqüentemente, a função de adequabilidade, *F[J(e,u)]*, a ser maximizada pelo algoritmo de evolução Lamarckiana é dada por

$$f[J(u,e)] = \frac{k}{1 + J(u,e)} \qquad (29)$$

onde k=300000. Quanto maior o valor de $f[J(u,e)]$, melhor o membro da população.

Tabela 1: Resultados preliminares obtidos com a otimização do *GPC* usando *OCP* (adota-se a partícula com melhor $J(u,e)$ após a realização de 10 experimentos).

tipo de *OCP*	$J(u,e)$ média	$J(u,e)$ máximo	$J(u,e)$ mínimo	$J(u,e)$ desvio padrão
1	0,4317	0,5845	0,0915	0,2156
2	0,4096	0,5828	0,0996	0,1951
3	0,3577	0,5547	0,1668	0,1792
4	0,3524	0,5801	0,1195	0,2197
5	0,4446	**0,5847**	0,1453	0,1813
6	0,3409	0,5738	0,1299	0,1940
7	0,3595	0,5723	0,1668	0,2079
8	0,3929	0,5757	0,2245	0,1714
9	**0,5161**	0,5791	**0,3354**	**0,1025**

Os melhores parâmetros obtidos para o *GPC* foram N_u = 6, N_2 = 6 e Γ = 0,11323 com a utilização da *OCP* do tipo 5 com $J(u,e)$ = 0,5847. Entretanto, a *OCP* do tipo 9 (com equação (26) usando distribuição de Gauss, *Gd*(), na parte "social" e distribuição de Cauchy na parte "cognitiva"), foi a que apresentou melhores resultados preliminares quanto a convergência média (maior valor) de $J(u,e)$ e menor desvio padrão de $J(u,e)$ entre os experimentos realizados.

Neste caso, os resultados do *GPC* quanto ao comportamento servo são apresentados na figura 3. Para análise deste comportamento utilizou-se as referências $y_r(t)$ = 0,70 (amostras 1 a 100) e $y_r(t)$= 0,50 (amostras 101 a 200).

Os parâmetros iniciais do estimador dos *MQR* na estimativa considerando-se um modelo matemático linear de segunda ordem são $a_1=a_2=b_1=b_2=0{,}10$. O traço inicial da matriz de covariância é $1000I_4$ para o projeto do *GPC*.

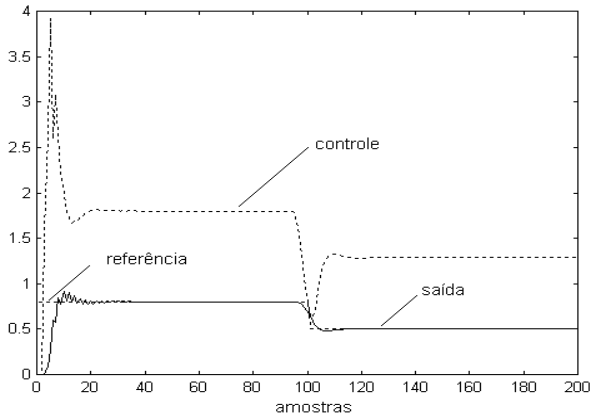

Figura 3: Simulação do projeto do *GPC* usando *OCP* do tipo 5 para análise do comportamento servo.

Os resultados apresentados na figura 3 mostram que o *GPC* apresentou comportamento servo com pequena variação da ação de controle para as duas primeiras referências.

5. Conclusão

Diversas metodologias de projeto do *GPC* têm sido estudadas, mas as técnicas possuem em comum a estrutura básica de projeto. As principais diferenças são relativas a forma em que as previsões futuras são calculadas, o tipo de modelo matemático utilizado e como a função custo é definida.

Neste artigo foram apresentadas abordagens de otimização usando *OCP* dos horizontes de controle, horizonte de previsão e ponderação da ação de controle de um *GPC* adaptativo. O estudo de caso abordado foi um processo não-linear do tipo Hammerstein.

Os resultados obtidos para o comportamento servo foram satisfatórios. No entanto, alguns estudos posteriores são necessários. Neste contexto, os trabalhos futuros serão ligados ao aprimoramento do *GPC* visam abordar: (i) estudos de projeto de *GPC* usando modelos matemáticos não-lineares para previsão do comportamento servo e regulatório; (ii) o desenvolvimento de metodologias que permitam otimização de parâmetros inteiros (N_u e N_2); (iii) otimização em tempo real; e (iv) aprimoramento e um estudo estatístico detalhado das novas abordagens de *OCP* propostas neste artigo.

Referências

[1] N. Bonavita and R. Tomasi. Improvements in process control through model-based techniques: a control system vendor's perspective. *IEEE Int. Conf. on Control Applications*, Trieste, Itália, pages 298-303, 1998.

[2] S. J. Qin and T. J. Badgwell. An overview of model predictive control technology. In: Kantor, J. C.; Garcia, C. E.; Carnahan, B. (eds.), *Proc. of the 5th International Conference on Chemical Process Control*, Tahoe, CA, AIChE Symp. Series 316, Vol. 93, pages 232-256, 1997.

[3] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers and Chemical Engineering*, 23: 667-682, 1999.

[4] M. Mahfouf, D. A. Linkens and M. F. Abbod. Multi-objective genetic optimisation of GPC and SOFLC tuning parameters using a fuzzy–based ranking method, *IEE Proc.-Control Theory Appl.*, 147(3): 344-354, 2000.

[5] V. Goggos and R. E. King. Evolutionary predictive control (EPC), *Computers Chem. Eng.*, 20: S817-S822, 1996.

[6] M. Martínez, J. Senté and X. Blasco. A comparative study of classical vs genetic algorithm optimization applied in GPC controller. *Proc. of the IFAC 13th Triennial World Congress*, San Francisco, USA, pages 327-332, 1996.

[7] L. S. Coelho and A. A. R. Coelho. Multivariable predictive control based on neural network model and simplex-evolutionary hybrid optimization. *Soft Computing in Industrial Applications*, Suzuki, Y.; S. Ovaska and T. Furuhashi; R. Roy and Y. Dote (eds.), Springer, London, UK, 2000.

[8] D. W. Clarke, C. Mohtadi and P. S. Tuffs. Generalized predictive control — part i. the basic algorithm, *Automatica*, 23(2): 137-148, 1987.

[9] D. W. Clarke, C. Mohtadi and P. S. Tuffs. Generalized predictive control — part ii. extensions and interpretations, *Automatica*, 23(2): 149-160, 1987.

[10] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. *Proc. of the 6th Int. Symp. on Micro Machine and Human Science*, Nagoya, Japan, Piscataway, NJ. IEEE Press, pages 39-43, 1995.

[11] J. Kennedy, J. and R. C. Eberhart. Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks IV*, Piscataway, NJ, IEEE Press, pages 1942–1948, 1995.

[12] R. C. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. *Proceedings of Congress on Evolutionary Computation*, Seoul, Korea. Piscataway, NJ, IEEE Press, 2001.

[13] D. W. Clarke and P. J. Gawthrop. Self-tuning controller. *IEE Proceedings*, 112(9): 929-934, 1975.

[14] J. E. S. Santos. Critérios de desempenho e aspectos de robustez na síntese de controladores preditivos adaptativos. *Dissertação de mestrado*, Engenharia Elétrica, UFSC, Florianópolis, SC, 1998.

[15] P. E. Wellstead and M. B. Zarrop. *Self-tuning systems: control and signal processing*. Chichester: Wiley, 1991.

[16] R. A. Krohling, L. S. Coelho and Y. Shi. Cooperative particle swarm optimization for robust control system design. *7th Word Conference on Soft Computing in Industrial Applications*, Granada, Spain, On-line, *http://site: decsai.ugr.es/wsc7*, 2002.

[17] Y. Shi and R. C. Eberhart. A modified particle swarm optimizer. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69-73. Piscataway, NJ: IEEE Press, 1998.

[18] Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimization. *Proceedings of the Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Service Center, pages 1945-1950, 1999.

[19] X. Yao and Y. Liu. Fast evolutionary programming. *Proceedings of 5th Annual Conference on Evolutionary Programming*, San Diego, CA, pages 451-460, 1996.

[20] K. Chellapilla. Combining mutation operators in evolutionary programming. *IEEE Trans. on Evolutionary Computation*, 2(3): 91-96, 1998.

[21] G. Rudolph. Local convergence rates of simple evolutionary algorithms with Cauchy mutations. *IEEE Trans. on Evolutionary Computation*, 1: 249-258, 1997.

[22] E. Katende, A. Jutan and R. Corless. Quadratic nonlinear predictive control, *Ind. Eng. Chem. Res.*, 37: 2721-2728.

[23] S. Drogies and D. De Geest. Predictive control: propositions for the design methodology, *American Control Conference*, San Diego, CA, pp. 647-651, 1999.

[24] K. Y. Rani and H. Unbehauen. Study of predictive controller tuning methods, *Automatica*, 33(12): 2243-2248, 1997.