UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ANÁLISE E IMPLEMENTAÇÃO DE ALGORÍTMOS DE CONTROLE ÓTIMO PELO MÉTODO DO GRADIENTE CONJUGADO.

BENEDITO RENE FISCHER
ORIENTADORES:
PROF.DR. CELSO PASCOLI BOTTURA
PROF.DR. MARIO JINO

DISSERTAÇÃO DE MESTRADO APRESENTADA À FACULDADE DE ENGENHARIA DA UNICAMP.

ABRIL - 1983

Dedico este trabalho

a meu pai .

A B S T R A C T

In this work a FORTRAN program to solve optimal control problems by the conjugate gradient method was developed. The program can be applied in a generic space of control inputs and it can deal with inequality constraints both in the control and input variables. The application of the method to several examples is made to include the main control input spaces.The results are compared with data supplied on the literature in order to appraise the performance of the algorithm.

R E S U M O

Neste trabalho foi desenvolvido um programa FORTRAN para a solução dos problemas de controle ótimo, pelo método do gradiente conjugado. O programa pode ser aplicado em um espaço de entradas de controle genérico, e pode lidar com restrições de desigualdade em ambas as variáveis, de entrada e de controle. A aplicação do método é feita a vários exemplos, para incluir os principais espaços das entradas de controle. De forma a avaliar o desempenho do algorítmo, os resultados são comparados com dados fornecidos na literatura.

Í N D I C E

"O saber não está na ciência alheia, que se absorve, mas, principalmente, nas idéias próprias, que se geram dos conhecimentos absorvidos, mediante a transmutação, por que passam, no espírito que os assimila. Um sabedor não é armário de sabedoria armazenada, mas transformador reflexivo de aquisições digeridas."

Rui Barbosa

C A P Í T U L O 1

INTRODUÇÃO

Com o avanço tecnológico das últimas décadas, tornou-se crítica a demanda de soluções precisas para um grande número de problemas de controle, onde os métodos clássicos (1,2,3), embora em certos casos extremamente úteis e práticos, não podem ser aplicados devido sua natureza ainda um tanto empírica. Tais problemas chamados de *controle ótimo*, envolvem a determinação de sinais de controle de forma a levar um processo a satisfazer uma série de restrições físicas e ao mesmo tempo minimizar (ou maximizar), algum critério de desempenho. A solução analítica de tal tipo de problema , em sua forma mais geral, é muito difícil e em certos casos inviável. Felizmente o próprio avanço tecnológico fechando o ciclo, nos fornece uma ferramenta poderosa, o computador, para a solução destes problemas.

Este trabalho tem por objetivo o desenvolvimento de um programa geral, chamado GRAMIN, para a solução numérica dos problemas de controle ótimo. O programa é baseado na generalização do método do gradiente conjugado para o espaço de entradas de controle, podendo trabalhar com os seguintes espaços: funções contínuas, entradas amostradas, conjunto de parâmetros, modulação por largura de pulso, e espaço do usuário, no qual a própria pessoa define as propriedades do espaço de controle que deseja trabalhar. É possível ainda , em cada um destes espaços, obter o controle ótimo, quando a variável de controle está sujeita à restrições de saturação. A codifica

ção do GRAMIN em Fortran IV, foi realizada de forma que o mesmo pudesse ser executado em diferentes computadores, com o mínimo de alterações. Ele foi testado no computador DEC-SYSTEM 10 da UNICAMP, no IBM-SYSTEM 370/145 e no HP-2100, ambos da UFSCAR, que são máquinas diferentes, com diferentes compiladores e palavras de 36,32 e 16 bits respectivamente. Em cada um destes computadores, o programa requereu apenas a mudança dos números que definem as unidades lógicas de entrada e saída.

Pretende-se com este trabalho, que o usuário à princípio sem o conhecimento das técnicas de Programação Matemática e com conhecimentos básicos da linguagem Fortran e da teoria de controle, tenha condições de se utilizar do programa para a resolução de seus problemas. Desta forma no capítulo 2 é feita uma revisão completa da teoria, de maneira que o usuário possa aplicar o programa em outros espaços de controle de seu interesse. Nesta revisão, introduz-se a Programação Matemática e os métodos de: gradiente ótimo, direções conjugadas, Davidon-Fletcher-Powell, e o método do gradiente conjugado para a minimização de funções no R^n. É apresentada a forma para a obtenção do gradiente em um espaço de controle genérico e também nos espaços usados pelo GRAMIN. Em seguida as técnicas de funções de Penalidades e Tranformação, para resolver o problema de controle ótimo com restrições nas variáveis de estado e controle, são discutidas. Finalmente é apresentado o algorítmo de Pagurek, para a solução dos casos de saturação na variável de controle.

No capítulo 3, utilizando-se dos algorítmos desenvolvidos no capítulo anterior, é detalhada a implementação do GRAMIN. É descrita ainda, a forma de uso do programa, bem como as subroutinas que o usuário deve fornecer, quando do uso do GRAMIN, na solução de seus problemas de controle.

O capítulo 4 trata da aplicação do GRAMIN à solução de três problemas de controle ótimo. No primeiro, são obtidas as entradas de controle de um motor DC, considerando que o controle pertence aos espaços de funções contínuas, entradas amostradas e modulação por largura de pulso. Nos espaços de funções contínuas e de modulação por largura de pulso, a sensitividade do método é testada quanto ao tipo de busca unidimensional empregada e quanto à variação de alguns parâmetros, considerados importantes para o desempenho do algorítmo. Ainda no espaço de modulação por largura de pulso, o algorítmo de Pagurek é analisado, e é testado também um controle com pulsos positivos. No segundo problema é feita a determinação dos parâmetros ótimos de um compensador, neste caso as entradas de controle pertencem ao espaço de parâmetros. Estes dois problemas abordam os quatro espaços de controle em que o GRAMIN se aplica diretamente, e visando uma análise do desempenho do programa, seus resultados são comparados com os de Hasdorff(28). O terceiro problema ilustra a versatilidade do algorítmo, através da estimação de parâmetros de um sistema de equações diferenciais ordinárias, em uma aplicação de interêsse na Engenharia Química.

Finalmente no capítulo 5 é discutido o uso do GRAMIN em microcomputadores, visando o controle digital direto de processos.

C A P Í T U L O 2

O MÉTODO DO GRADIENTE CONJUGADO APLICADO ÀS SOLUÇÕES DO PROBLEMA DO CONTROLE ÓTIMO.

O objetivo deste capítulo é desenvolver a formulação matemática para a obtenção de algorítmos, visando a solução do problema de controle ótimo em sua forma mais geral. Nesta forma, a equação de estado do sistema, seu funcional de custo e as restrições sobre o mesmo, podem ser definidos por equações não lineares. O problema pode se apresentar com restrições tanto nas variáveis de estado como nas de controle.

O método básico utilisado para a obtenção de tais algorítmos, é o *método do gradiente conjugado*. Ele é desenvolvido inicialmente para a minimização de funções no R^n , sendo em seguida generalizado para a solução do problema de controle ótimo irrestrito. Finalmente são apresentadas modificações nos algorítmos que permitem estender o tratamento para os casos com restrições.

2.1 - O PROBLEMA DE CONTROLE ÓTIMO.

O primeiro passo no desenvolvimento da formulação matemática para a obtenção dos algorítmos de controle ótimo, é a descrição, por meio das equações de estado, da dinâmica do sistema a ser controlado. Para um sistema contínuo no tempo, as equações se apresentam como um conjunto de equações diferenciais de primeira ordem do tipo:

$$\frac{d\underline{x}(t)}{dt} = \underline{f}(\underline{x}(t),\underline{u}(t)) \qquad (2.1.1)$$

onde $\underline{x}(t)$ é um vetor de nxl componentes, representando o estado do sistema no tempo t; $\underline{u}(t)$ é um vetor de mxl componentes que especifica a entrada do sistema, e $\underline{f}$ é uma função vetorial de nxl componentes.

Sendo dadas as equações de estado, um conjunto de condições de contôrno sobre as variáveis de estado nos instantes de tempo inicial e final, e um conjunto de vínculos sobre as variáveis de estado e de controle, o *problema de controle ótimo* (4,5,6,7) pode ser enunciado como:

"*Determine um controle admissível* $\underline{u}(t)$, *de forma a minimizar (ou maximizar) um dado critério de desempenho ou função de custo*".

A função de custo é descrita como um escalar do tipo:

$$J = G(\underline{x}(t_f),\ t_f) + \int_{t_0}^{t_f} F(\underline{x}(t),\underline{u}(t),t)dt \qquad (2.1.2)$$

sendo $x(t_0)$ e $S(x(t_f))$ as condições de contôrno sobre as variáveis de estado, onde S é chamado de conjunto objetivo e t_0 e t_f são o instante inicial e final respectivamente.

Existem várias teorias utilizadas para resolver o problema de controle ótimo, as principais são:

1 - Cálculo de Variações (4,7-9);

2 - Princípio do Máximo (4,6,7,10,11);

3 - Programação Dinâmica (4,6,12-14);

4 - Quasilinearização (6);

5 - Embutimento Invariante (Invariant Imbedding),(6);

6 - Programação Matemática (4).

Os algorítmos apresentados e desenvolvidos neste trabalho, pertencem a um ramo da Programação Matemática. Sendo assim, a próxima seção trata tal teoria com mais destalhes.

2.2 - PROGRAMAÇÃO MATEMÁTICA.

A aplicação das técnicas de Programação Matemática na soluções de problemas de controle ótimo, iniciou-se por volta de 1960 (4). A Programação Matemática, a exemplo das teorias citadas na seção anterior, procura resolver alguns casos de otimização. Tais problemas envolvem a obtenção do máximo ou mínimo de uma função , de uma ou mais variáveis, sujeitas a certas restrições.

O emprego das técnicas de Programação Matemática apresenta as seguintes vantagens (4):

1 - Capacidade de manipular eficientemente os vínculos de desigualdade no controle e nas variáveis de estado, onde outros métodos tais como: Cálculo de Variações e Princípio do Máximo, poderiam levar a um problema com condições de contorno em dois pontos extremos ("Two-Point Boundary-Value Problem", TPBVP), de difícil solução.

2 - Não requerem tanta memória do computador, como as técnicas de Programação Dinâmica.

Apesar destas vantagens, podem existir problemas particulares, para os quais os outros métodos sejam mais adequados. Entretanto, para uma grande classe de problemas de controle ótimo a

Programação Matemática é mais eficiente, sendo que em alguns casos ela fornece a única solução possível (4).

O problema de Programação Matemática é o de encontrar o valor de n variáveis $x_1, \ldots, x_n$ de forma a satisfazer:

$$g_i(x_1, \ldots, x_n) \geq 0 \ , \quad i = 1, \ldots, p \qquad (2.2.1)$$

$$h_j(x_1, \ldots, x_n) = 0 \ , \quad j = 1, \ldots, q \qquad (2.2.2)$$

e a minimizar ou maximizar:

$$\phi(x_1, \ldots, x_n) = J \qquad (2.2.3)$$

As equações (2.2.1) e (2.2.2) são chamadas *restrições* e ϕ é chamada *função objetivo*. Um ponto x* é dito ótimo, se ele satisfaz (2.2.1) a (2.2.3).

2.2.1 - OS RAMOS DA PROGRAMAÇÃO MATEMÁTICA.

Dependendo da forma da função objetivo e das restrições impostas às variáveis, os problemas de programação Matemática podem ser subdivididos em vários tipos , tais como: Programação Linear , Quadrática, Geométrica, Não Linear, Inteira, Estocástica e Dinâmica.

1- Programação Linear.

O problema será chamado de Programação Linear (15-21,26), se $f(\underline{x})$, $g(\underline{x})$ e $h(\underline{x})$, dados pelas equações (2.2.1) a (2.2.3), forem funções lineares da variável $\underline{x}$. Os problemas de Programação Linear podem ser eficientemente resolvidos pelo método simplex, ou

simplex revisado (15-20). Na solução de problemas de grande porte, que podem levar a matrizes com 10 000 linhas ou mais, as técnicas de partição e decomposição podem ser empregadas (17,22).

2 - Programação Quadrática.

Se $g(\underline{x})$ e $h(\underline{x})$ são funções lineares e a função objetivo é uma função quadrática de $\underline{x}$, o problema é dito de Programação Quadrática (18,19,22,23). Em Estatística problemas de mínimos quadrados sujeitos à restrições, frequentemente levam à Programação Quadrática. Os métodos para a solução de tais problemas são basicamente variações do método simplex.

3 - Programação Geométrica.

Na Programação Geométrica (4,19), a função objetivo tem a forma:

$$\phi(\underline{x}) = \sum_{i=1}^{n} C_i \prod_{j=1}^{n} (x_j)^{a_{ij}} \quad , \tag{2.2.4}$$

sujeita às restrições:

$$x_j > 0 \qquad j = 1,2, \ldots , n \tag{2.2.5}$$

onde C_i = coeficientes constantes e positivos,

a_{ij} = constantes reais,

x_j = variáveis do problema.

Neste caso, a solução do problema pode ser obtida pela solução de um conjunto de equações lineares (19).

4 - Programação Não Linear.

Um tipo de programação mais geral, que engloba os casos anteriores, é a Programação Não Linear (4,17,18,26,22-24). O problema é considerado de Programação Não Linear, caso ocorra pelo menos uma não linearidade nas restrições ou na função objetivo. Neste caso, as funções ϕ, $\underline{g}$ e $\underline{h}$ definidas em (2.2.1) a (2.2.3) são as mais gerais possíveis. As técnicas de Programação Não Linear, podem ser empregadas, portanto, nos casos de Programação Linear, Geométrica e Quadrática. Entretanto, os algorítmos específicos para cada caso, são mais eficientes que os de Programação Não Linear , quando aplicados a estes casos.

A solução do problema de controle ótimo, frequentemente exige a solução de um problema de otimização, em que tanto o funcional de custo, quanto as restrições são não lineares. Nestes casos os métodos de Programação Não Linear podem ser aplicados com sucesso (4, 5,27-36). Por este motivo, na seção 2.3 serão abordados os principais métodos e algorítmos de Programação Não Linear.

5 - Programação Inteira.

A Programação Inteira é uma técnica especial para a resolução de problemas de Programação Matemática, em que as variáveis podem assumir apenas valores inteiros (18,22,23). Apesar da maioria dos trabalhos nesta área serem voltados para a Programação Linear Inteira, existem métodos para a solução dos casos não lineares. Um caso particular da Programação Inteira se apresenta na Programação Mista, em que parte das variáveis assumem valores inteiros e parte pode assumir valores reais quaisquer.

6 - Programação Estocástica.

Caso as variáveis tenham natureza aleatória, o problema é chamado de Programação Estocástica (4,22).

7 - Programação Dinâmica.

Alguns autores (18,22,23), consideram a Programação Dinâmica, citada na seção anterior (2,1) como um problema especial de Programação Matemática.

2.3 - PROGRAMAÇÃO NÃO LINEAR - MÉTODOS DE OTIMIZAÇÃO IRRESTRITA.

Os problemas de Programação Não Linear podem se apresentar sob duas formas distintas. A forma mais geral, chamada Programação Não Linear Restrita (22,37), trata do problema de otimização de equações do tipo (2.2.3), sujeitas à restrições impostas por (2.2.1) e (2.2.2). Na ausência destas restrições, tem-se o caso da Programação Não Linear Irrestrita.

Nos métodos de otimização irrestrita (22,37,38,39), busca-se o ótimo de uma função de n variáveis, que não está sujeita à restrições. Neste caso busca-se um valor ótimo J*, para a equação (2.2.3) e não se tem equações do tipo (2.2.1) e (2.2.2). O valor ótimo pode ser um máximo ou um mínimo.

O método do gradiente conjugado é uma das técnicas de otimização irrestrita, em particular neste trabalho nos limitaremos ao *problema de minimização irrestrita*. Além disso, alguns dos métodos de otimização restrita, convertem o problema com restrições em um problema irres

trito, como será mostrado na seção 2.5 .

Os métodos de otimização irrestrita podem ser divididos em duas classes:

a) métodos que se utilizam das derivadas da função objetivo (36-39) e;

b) métodos que não as usam, chamados métodos diretos (36-42).

Os métodos que empregam o gradiente e (ou) derivadas segundas, convergem mais rapidamente que aqueles diretos. Entretanto , em problemas com um grande número de variáveis, pode se tornar difícil a obtenção do gradiente ou derivadas segundas, quer por meio de uma expressão algébrica ou por métodos numéricos. Em tais casos melhores resultados podem ser obtidos pelos métodos diretos (36-42). Na prática a escolha de um método vai depender do problema considerado, da velocidade de convergência, confiabilidade, precisão do método e experiência do usuário.

Nos próximos parágrafos serão apresentados alguns métodos pertencentes ao caso (a), com ênfase no método do gradiente conjugado. Ele é importante porque constitui a principal ferramenta para a obtenção dos algorítmos de controle ótimo, desenvolvidos neste trabalho (capítulo 3).

2.3.1 - O MÉTODO DO GRADIENTE ÓTIMO.

O método do gradiente é um dos métodos mais intuitivos, empregados para a obtenção do mínimo de um funcional $\phi(\underline{x})$. O gradiente calculado em um ponto $\underline{x}_0$, fornece a direção de maior crescimento do funcional em $\underline{x}_0$, logo o sentido contrário ao do gradiente resulta na maior diminuição do valor do funcional.

Um algorítimo de minimização pelo método do gradiente é expresso como:

$$\underline{x}_{k+1} = \underline{x}_k - \lambda_k \underline{g}(\underline{x}_k) \qquad (2.3.1)$$

onde $\underline{g}(\underline{x}_k)$ é o gradiente da função objetivo ϕ em $\underline{x}_k$, e λ_k é um escalar que determina o passo a ser dado na iteração k, na direção determinada pelo gradiente.

O algorítmo descrito por (2.3.1), nem sempre atinge o ponto de mínimo x* em um único passo, e várias iterações devem ser realizadas para se chegar ao mínimo. O escalar λ , que a cada iteração determina o comprimento do passo, pode ser obtido de várias maneiras, podendo inclusive ser uma constante unitária. Se λ é escolhido de forma a minimizar a função objetivo a cada iteração, desta forma satisfazendo a equação:

$$\frac{d\underline{f}(\underline{x}_k - \lambda_k \underline{g}(\underline{x}_k))}{d\lambda} = 0 \qquad (2.3.2)$$

o método é dito do gradiente com passo ótimo, ou do *gradiente ótimo*. Métodos analíticos ou numéricos podem ser empregados para a determinação de λ , de forma a satisfazer a equação (2.3.2). O método numérico empregado neste trabalho, está descrito no apêndice A1.

O método do gradiente ótimo, segundo Himmeblau (37), quando aplicado a uma função objetivo com diferenciabilidade de terceira ordem, converge no limite quando $k \to \infty$. Contudo, em situações práticas, nem sempre é necessário ter um número infinito de iterações para a obtenção de uma razoável aproximação do mínimo. Porém se a função objetivo, próximo do mínimo, tiver a forma de um vale estreito e alongado, o método do gradiente ótimo apresentará um comportamento oscilatório, resultando em uma convergência por de-

mais lenta, inviabilisando seu uso em casos práticos.

2.3.2 - O MÉTODO DAS DIREÇÕES CONJUGADAS.

Os problemas de comportamento oscilatório e convergência lenta, característicos do método do gradiente ótimo, podem ser evitados pelo uso de métodos com convergência quadrática, e que levem em consideração informações à respeito da curvatura da função objetivo (44). Convergência quadrática significa, que o mínimo de uma função de n variáveis independentes é localizado em um numéro finito de iterações, geralmente em n passos, se a função objetivo for quadrática.

O método das direções conjugadas (26,28,37,44), apresentado neste parágrafo possui tais propriedades.

Para introduzir este método, vamos considerar que a função objetivo $\phi(\underline{x})$ possa ser aproximada por uma função $F(\underline{x})$, como resultado de uma expansão em série de Taylor em torno do ponto de mínimo $\underline{h}$, com têrmos até de segunda ordem. Neste caso:

$$\phi(\underline{x}) \simeq F(\underline{x}) = F(\underline{h}) + \frac{1}{2} <(\underline{x} - \underline{h}), A(\underline{x} - \underline{h})> \qquad (2.3.3)$$

onde A é o operador derivada segunda calculado no ponto de mínimo.

Se o operador linear A é simétrico e definido positivo, então $F(\underline{x})$ possui um único mínimo (28,44). Este mínimo é determinado pelo método das direções conjugadas da seguinte forma: partindo-se de uma estimativa inicial $\underline{x}_0$ do ponto de mínimo, é construída uma sequência,

$$\{\underline{x}_i\} = [\underline{x}_0, \underline{x}_1, \cdots, \underline{x}_i, \cdots] \qquad (2.3.4)$$

onde

$$\underline{x}_{i+1} = \underline{x}_i + \alpha_i \underline{p}_i \qquad i = 0,1,2, \ldots , n-1 \qquad (2.3.5)$$

Nesta expressão α_i é escolhido por meio de uma busca unidimensional de forma a minimizar $F(\underline{x}_i+\alpha\ \underline{p}_i)$ a cada iteração. Em outras palavras α_i satisfaz:

$$\frac{dF(\underline{x}_i + \alpha\ \underline{p}_i)}{d\alpha}\Big|_{\alpha=\alpha_i} = F'(\underline{x}_i + \alpha_i \underline{p}_i) \cdot \underline{p}_i = 0 \qquad (2.3.6)$$

Se $\underline{g}(\underline{x}_{i+1})$ é o gradiente da função $F(\underline{x})$ no ponto $\underline{x}_{i+1}$, segue da definição de gradiente de um funcional (28), e da equação (2.3.6) que:

$$\langle \underline{g}(\underline{x}_{i+1}), \underline{p}_i \rangle = 0 \qquad (2.3.7)$$

As direções de busca $\underline{p}_i$ são direções A conjugadas. Por direções A conjugadas queremos dizer, que os vetores $\underline{p}_i$ são conjugados com relação ao operador linear A, ou seja:

$$\begin{aligned} &\langle \underline{p}_i, A\underline{p}_j \rangle = 0 \qquad \text{se} \quad \underline{p}_i = \underline{p}_j \\ &\langle \underline{p}_i, A\underline{p}_j \rangle \neq 0 \qquad \text{se} \quad \underline{p}_i \neq \underline{p}_j \end{aligned} \qquad (2.3.8)$$

As referências (26,28,37,44), mostram que o método descrito pela equação (2.3.5), satisfazendo (2.3.6) e (2.3.8), fornece o mínimo de uma função quadrática em pelo menos n passos. Para funções mais gerais, à medida que a variável iterada se aproxima do valor ótimo, a função pode ser melhor descrita por uma expansão quadrática (como a equação (2.3.3)), e a convergência pode ser

obtida. Entretanto em tais casos o número de passos necessários para a convergência é maior do que n (44).

A geração do conjunto de direções A conjugadas torna-se problemática, quando a função $F(\underline{x})$ e o gradiente $\underline{g}(\underline{x})$ são definidos numéricamente, de forma que o operador linear A não é obtido explicitamente. Para resolver este problema, foram desenvolvidos métodos como os de: Davidon-Fletcher-Powell (36,37), o das Tangentes Paralelas (36,37) e o método do gradiente conjugado (28,44). Todos eles são métodos de direções conjugadas, diferindo apenas na forma como as direções são calculadas.

Box (45) , comparou oito métodos diferentes para a minimização de funções com duas, três, cinco, dez e vinte variáveis. Neste trabalho foram analisados tantos os métodos diretos como os indiretos, inclusive os métodos indiretos de Davidon-Fletcher-Powell e o método do gradiente conjugado . Segundo Box, dentre estes métodos, o que apresentou maior eficiência para a minimização de funções foi o de Davidon-Fletcher-Powell.

O método Davidon-Fletcher-Powell (47) é uma reformulação do método original de Davidon (46), proposto por Fletcher e Powell . Neste método, as direções A conjugadas $\underline{p}_i$ são dadas por:

$$\underline{p}_i = -H_i \underline{g}_i \qquad (2.3.9)$$

onde H_i para i = 1,2,3,... é uma sequência de matrizes simétricas definidas positivas, geradas por:

$$H_{i+1} = H_i + \frac{\underline{p}_i \underline{p}_i^{\dagger}}{\underline{p}_i^{\dagger} A \underline{p}_i} - \frac{H_i \underline{\gamma}_i \underline{\gamma}_i^{\dagger} H_i}{\underline{\gamma}_i^{\dagger} H_i \underline{\gamma}_i} \qquad (2.3.10)$$

com $$\underline{\gamma}_i = \underline{g}_{i+1} \qquad (2.3.11)$$

e H_0 é uma matriz arbitrária positiva definida, quase sempre a matriz identidade.

A aplicação deste método requer a armazenagem da matriz H, que computacionalmente pode representar um inconveniente, podendo mesmo tornar proibitivo o seu uso, já que dependendo do número de variáveis, há a necessidade de se reservar um grande espaço na memória para a armazenagem de H, aumentando também o tempo de processamento. Por exemplo, para a utilização do método de Davidon-Fletcher-Powell nos algorítmos do capítulo 3, a matriz H deveria ser dimensionada com 40401 elementos (201 X 201). Desta forma é desejável a existência de um método, que mesmo sem apresentar tal eficiência, mantenha a característica dos métodos de direções conjugadas e seja operacionalmente mais econômico. O método do gradiente conjugado, descrito na próxima seção, apresenta tais características.

2.3.3 - O MÉTODO DO GRADIENTE CONJUGADO.

O *método do gradiente conjugado* foi desenvolvido originalmente por Hestenes e Stiefel(48), para a solução em n passos de um conjunto de equações lineares simultâneas do tipo:

$$\bar{A}\underline{x} = \underline{b} \qquad (2.3.12)$$

tendo uma matriz de coeficientes simétrica e positiva definida. A solução de tal conjunto de equações fornece um vetor $\underline{x}$ tal que:

$$\underline{g}(\underline{x}) = \bar{A}(\underline{x} - \underline{h}) = 0 \qquad (2.3.13)$$

com

$$\underline{b} = \bar{A}\,\underline{h}\,. \qquad (2.3.14)$$

A equação (2.3.13), é exatamente a condição para que $\underline{x}$ seja o mínimo de (2.3.3).

O algorítmo do gradiente conjugado pode ser aplicado em funções gerais, que não são quadráticas (21,28). Neste caso, não se garante que a convergência é obtida em n passos, e sim como resultado de um processo iterativo. O algorítmo em sua forma geral(26, 28,36,37,44), pode ser descrito pelo seguinte conjunto de equações:

$$\underline{p}_0 = -\underline{g}(\underline{x}_0) = -\underline{g}_0 \tag{2.3.15}$$

$$\underline{x}_{i+1} = \underline{x}_i + \alpha_i \underline{p}_i \tag{2.3.16}$$

onde $\alpha_i > 0$ minimiza $F(\underline{x}_i + \alpha_i \underline{p}_i)$, (2.3.17)

$$\underline{p}_{i+1} = -\underline{g}_{i+1} + \beta_i \underline{p}_i \tag{2.3.18}$$

com $$\beta_i = \frac{\langle \underline{g}_{i+1}, \underline{g}_{i+1} \rangle}{\langle \underline{g}_i, \underline{g}_i \rangle} \tag{2.3.19}$$

Na equação (2.3.15), x_0 é uma estimativa inicial arbitrária do mínimo. Em (2.3.17), α_i é obtido por meio da busca unidimensional descrita no apêndice A1.

Beckman no capítulo 4 da referência (21), mostra que o algorítmo descrito pelas equações (2.3.15) a (2.3.19), é um método de direções conjugadas e particularmente converge em n passos, quando a função F é quadrática.

Hasdorff (28) também demonstra, que o método acima é de direções conjugadas e prova que o algorítmo apresenta convergência quadrática, mesmo quando aplicado a uma função genérica.

Um problema, no entanto, pode ocorrer com a razão de convergência do algorítmo do gradiente conjugado, quando o mesmo é aplicado à funções com curvas de níveis de características seme

lhantes às da função banana de Rosenbrock (40), veja figura 2.1. Neste caso as direções A conjugadas $\underline{p}_i$, são aproximadamente paralelas (44), de forma que os pontos $\underline{x}_i$ não estão suficientemente separados, tornando muito lenta a convergência para o mínimo da função.

Pagurek (49) apresentou um método do gradiente conjugado, que a exemplo do método de Davidon-Fletcher-Powell, faz uso da derivada segunda da função objetivo, desta forma considerando a informação à respeito da curvatura da função próximo ao mínimo. Com este procedimento a velocidade de convergência para o mínimo é aumentada (44). Visando também uma melhoria na razão de convergência para o mínimo, Hasdorff (28) propos outro método, chamado método do gradiente conjugado escalonado, no qual o espaço do funcional de custo é transformado, de maneira que as curvas de níveis do funcional sejam aproximadamente esféricas, facilitando a busca do mínimo.

Contudo, tanto o algorítmo de Pagurek, como o do gradiente conjugado escalonado, apresentam desvantagens semelhantes àquelas do método de Davidon-Fletcher-Powell. O método de Pagurek requer a armazenagem do operador derivada segunda da função objetivo; e no gradiente conjugado escalonado deve-se armazenar a matriz de transformação do subespaço original para o subespaço escalonado. Em ambos os casos, a quantidade de memória de computador necessária é da mesma ordem de grandeza que a requerida pelo algorítmo de Davidon-Fletcher-Powell.

Fletcher e Reeves (44) propuseram uma solução mais simples para a busca do mínimo, quando a função objetivo apresentar características semelhantes às da função de Rosenbrock. O processo consiste em utilizar-se do método do gradiente conjugado, e reverter periódicamente a direção de busca $\underline{p}_i$ para a do gradiente ótimo

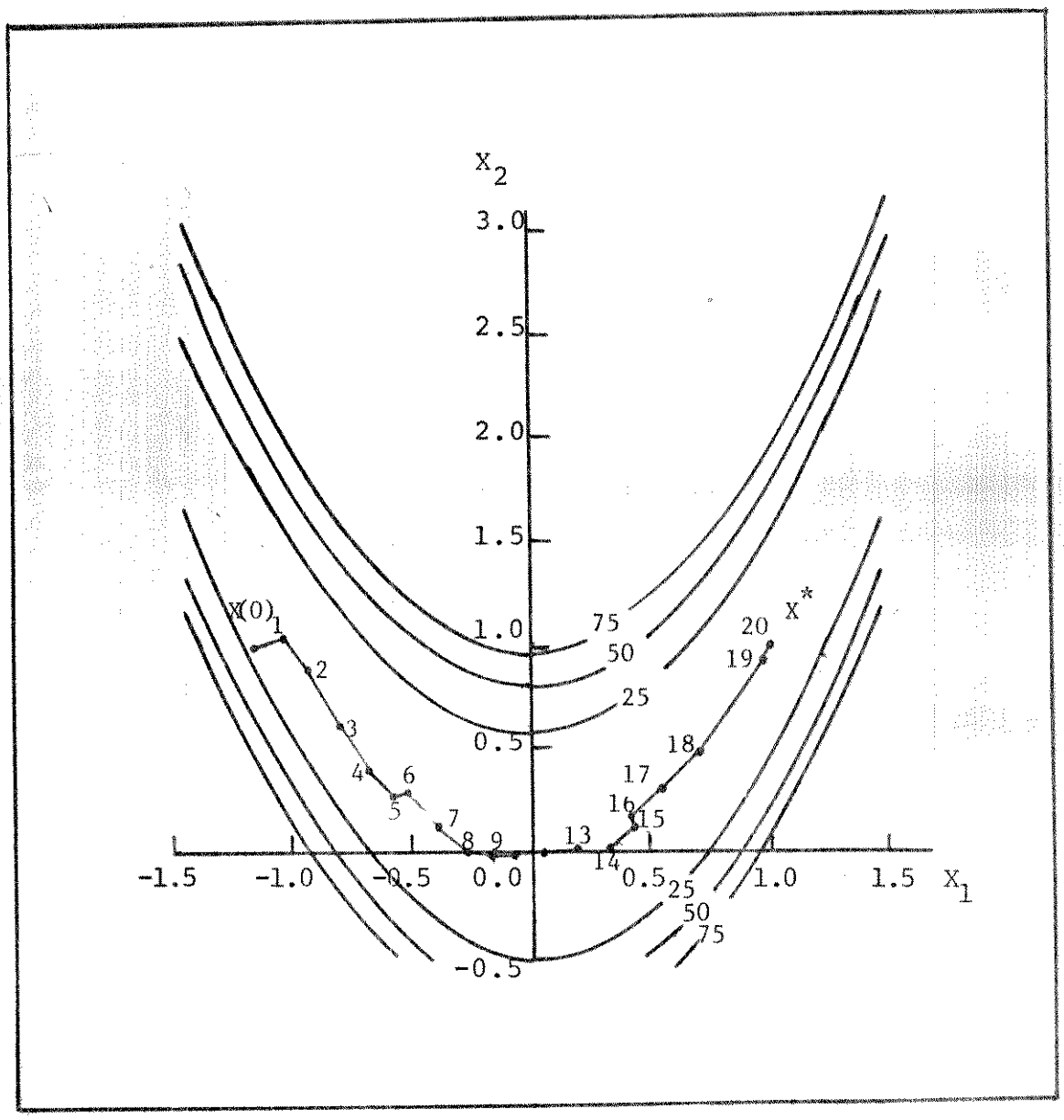


Fig. 2.1 - METODO DO GRADIENTE CONJUGADO APLICADO À MINIMIZAÇÃO DA FUNÇÃO BANANA DE ROSEMBROCK (37)

$-\underline{g}_i$, tal reversão é conseguida reinicializando o processo à partir do valor corrente de $\underline{x}_i$, e se descartando da experiência prévia, util ou não, transmitida pela direção $\underline{p}_i$. O processo mantém as propriedades de convergência quadrática, se as reinicializações não são mais frequentes que as n iterações. Fletcher e Reeves obtiveram os melhores resultados reinicializando o processo à cada n+1 iterações.

Este procedimento é o que foi adotado para melhorar a convergência dos algorítmos desenvolvidos no capítulo 3, quando aplicados aos problemas do capítulo 4.

2.4 - GENERALIZAÇÃO DO MÉTODO DO GRADIENTE CONJUGADO PARA A SOLUÇÃO DE PROBLEMAS DE CONTROLE ÓTIMO.

Até o presente momento, o desenvolvimento feito para o método do gradiente conjugado, se aplica à minimização de funções cujo domínio é o R^n, tratando-se neste caso, de otimização com dimensão finita (5). Entretanto para aplicações do método em problemas de controle ótimo, se faz necessário a extensão do mesmo para espaços mais gerais, de forma que as entradas de controle possam ser representadadas por funções contínuas, conjuntos de parâmetros, entradas amostradas, etc... Um espaço que engloba tais tipos de elementos é o espaço de Hilbert (28,50).

Lasdon, Mitter e Waren (27), foram os primeiros a extender o método do gradiente conjugado à solução de problemas de controle ótimo. Posteriormente Pagurek e Woodside (49), fizeram o mesmo para o método do gradiente conjugado de segunda ordem, discutido na seção 2.3.3 . Também outros autores, tais como Noton(5) e Hasdorff(28), apresentam a generalização do método para o espa

ço de funções de interesse nos problemas de controle ótimo.

A generalização apresentada nesta seção, segue a linha desenvolvida por Hasdorff(28), por ser uma das mais completas e detalhadas, que temos conhecimento.

2.4.1 - FORMA GERAL PARA OBTENÇÃO DO GRADIENTE DE FUNCIONAIS DE CUSTO DE SISTEMAS DE CONTROLE.

Com a generalização do método do gradiente conjugado, pretende-se utilizá-lo para a obtenção de entradas de controle que minimizem um certo critério de desempenho, ou funcional de custo, durante a operação do sistema. Portanto, para o emprego do método, o funcional de custo e seu gradiente devem ser calculados à partir de um dado argumento. A obtenção do valor do funcional é feita por substituição direta de seu argumento. Entretanto a obtenção do gradiente não é tão óbvia, e neste parágrafo será apresentado um método geral para a obtenção do gradiente dos funcionais de custo de interesse nos sistemas de controle.

Vamos considerar que F seja um funcional de um espaço de Hilbert H em R^1, como mostra a figura (2.2). Sejam $\underline{x}_0$, $\underline{t}$, e $\underline{z}$ elementos de H, sendo $\underline{t}$ e $\underline{z}$ definidos como:

$$\underline{t} = \varepsilon \underline{z} \tag{2.4.1}$$

onde

$$\|\underline{z}\| = 1 \tag{2.4.2}$$

e ε é um escalar qualquer. Da expansão por série de Taylor do funcional podemos escrever(51):

$$F(x_0 + \underline{t}) = F(\underline{x}_0) + < \underline{g}(\underline{x}_0), \underline{t} > + 1/2 < \underline{t}, A(\underline{x}_0)\underline{t} > + O^2(\underline{t}) \tag{2.4.3}$$

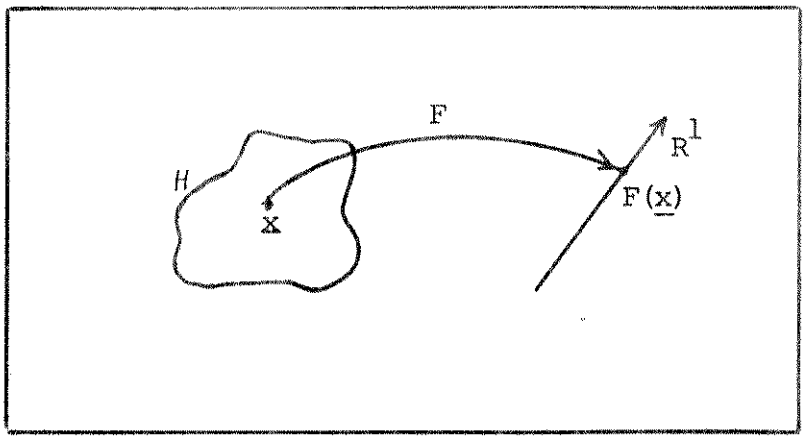


Fig 2.2 - Representação de um funcional F, de um espaço de Hilbert H em R^1 (28).

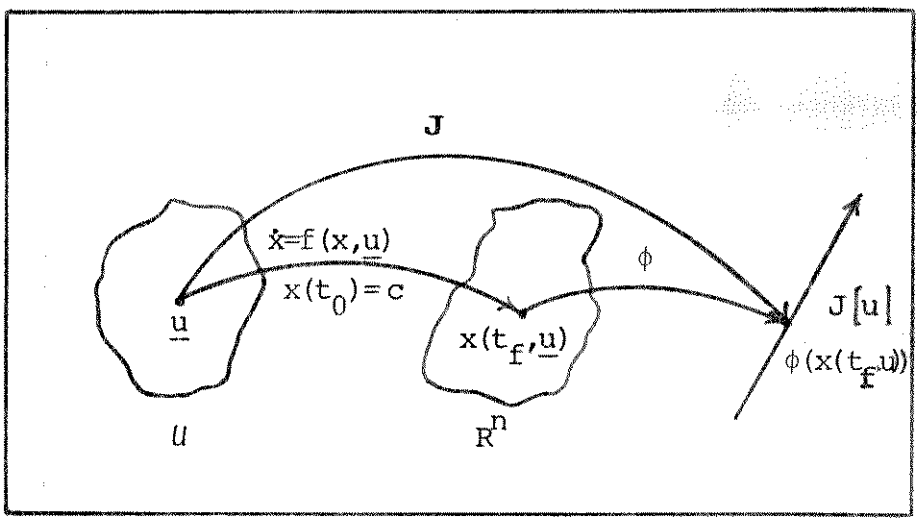


Fig. 2.3 - O Funcional de custo representado como um operador composto (28).

onde $\underline{g}(\underline{x}_0) \in H$ é o gradiente de F em $\underline{x}_0$, $A(\underline{x}_0)$ é o operador derivada segunda em $\underline{x}_0$ e $O^2(\underline{t})$ são os termos de ordens superiores. Substituindo (2.4.1) em (2.4.3) tem-se:

$$F(\underline{x}_0 + \varepsilon\underline{z}) = F(\underline{x}_0) + \varepsilon<\underline{g}(\underline{x}_0),\underline{z}> + 1/2\ \varepsilon^2<\underline{z},A(\underline{x}_0)\underline{z}> + O^2(\underline{t}) \quad (2.4.4)$$

Diferenciando ambos os membros de (2.4.4) em relação a ε e calculando em $\varepsilon = 0$, obtem-se:

$$\frac{d}{d\varepsilon} F(\underline{x}_0 + \varepsilon\underline{z})\Big|_{\varepsilon=0} = <\underline{g}(\underline{x}_0),\underline{z}> \quad (2.4.5)$$

O termo do lado esquerdo de (2.4.5) é chamado de diferencial de Gateaux(50). A equação (2.4.5), fornece o método básico para a obtenção do gradiente, que consiste em se realizar as operações definidas no lado esquerdo de (2.4.5), sobre o funcional F e depois identificar $\underline{g}(\underline{x}_0)$ usando a definição de produto escalar no espaço que se está trabalhando.

Para simplificar a manipulação das expressões, o problema de controle ótimo definido no parágrafo 2.1, será reformulado da seguinte maneira.

$$\frac{d\underline{x}(t)}{dt} = \dot{\underline{x}}(t) = \underline{f}(\underline{x}(t),\underline{u}(t)); \quad \underline{x}(t_0) = \underline{c} \quad (2.4.6)$$

com $\underline{x}(t) \in R^n$ e $\underline{u}(t) \in U$, onde U é um espaço de Hilbert genérico, $\underline{x}(t_0)$ é o estado inicial do sistema e $\underline{c}$ é um vetor constante. Da equação (2.4.6), pode-se observar que $\underline{f}$ é uma aplicação com domínio em $R^n \times U$ e codomínio em R^n. Considera-se que $\underline{f}$ tenha pelo menos derivadas parciais segundas, com relação a $\underline{x}$ e $\underline{u}$, contínuas e que para um dado $\underline{u}$, (2.4.6) tenha solução única no intervalo de tem

po $[t_0, t_f]$, ou seja , para uma dada entrada de controle $\underline{u}$, $\underline{x}(t_f, \underline{u})$ é única. Considerando a unicidade da equação (2.4.6), a equação (2.1.2) pode ser reformulada como:

$$J = J[\underline{u}] = \phi(\underline{x}(t_f, \underline{u})) \qquad (2.4.7)$$

Pode-se observar da equação (2.4.7), que o funcional de custo $J[\underline{u}]$ é uma função composta, pois dada uma entrada de controle $\underline{u}$ e $\underline{u}$ e $\underline{x}(t_f, \underline{u}) \in R^n$ que é obtido resolvendo (2.4.6), o funcional $J[\underline{u}] \in R^1$ é calculado por meio de $\phi(\underline{x}(t_f, \underline{u})$, como pode ser visto na figura (2.3) .

A exemplo do estabelecido no parágrafo 2.1, o problema do controle ótimo é encontrar $\underline{u}^* \in U$, que minimiza J. A diferença fundamental entre a formulação do parágrafo 2.1 e o estabelecido aqui é que neste caso não são impostas restrições às variáveis de estado e de controle, tratando-se portanto de um problema de minimização irrestrita, tornado viável o uso do método do gradiente conjugado .

Como já foi visto, a utilização do método requer o cálculo do funcional $J[\underline{u}]$, que é obtido da equação (2.4.7), e do gradiente $\underline{g}(\underline{u})$. De forma a identificar $\underline{g}(\underline{u})$, substituímos $J[\underline{u}]$ em (2.4.5) resultando:

$$\frac{dJ[\underline{u} + \varepsilon\underline{z}]}{d\varepsilon}\bigg|_{\varepsilon=0} = \frac{d\phi(\underline{x}(t_f, \underline{u} + \varepsilon\underline{z}))}{d\varepsilon}\bigg|_{\varepsilon=0} = \left\langle \nabla_{\underline{x}}\phi(\underline{x}(t_f, \underline{u} + \varepsilon\underline{z})), \frac{d\underline{x}}{d\varepsilon}(t_f, \underline{u} + \varepsilon\underline{z}) \right\rangle\bigg|_{\varepsilon=}$$

(2.4.8)

$$\text{Denotando} \quad \frac{d\underline{x}(t_f, \underline{u} + \varepsilon\underline{z})}{d\varepsilon}\bigg|_{\varepsilon=0} \quad \text{por} \quad \frac{d\underline{x}}{d\varepsilon}(t_f, \underline{u}) \qquad (2.4.9)$$

e usando a equação (2.4.5) tem-se:

$$\frac{dJ[\underline{u}+\varepsilon\underline{z}]}{d\varepsilon}\bigg|_{\varepsilon=0} = \langle g(\underline{u}),\underline{z}\rangle = \left\langle \nabla_x \phi(\underline{x}(t_f,\underline{u})), \frac{d\underline{x}}{d\varepsilon}(t_f,\underline{u}) \right\rangle \qquad (2.4.10)$$

permitindo a identificação do gradiente $\underline{g}(\underline{u})$, que para cada espaço de controle considerado, fica reduzido ao cálculo de $\frac{d\underline{x}}{d\varepsilon}$ dado por (2.4.9).

A seguir usando este procedimento, determinaremos as expressões para o gradiente nos seguintes espaços de entradas de controle:

1-funções contínuas,

2-conjunto de parametros,

3-entradas amostradas,

4-modulação por largura de pulso.

2.4.2- O GRADIENTE NO ESPAÇO DE FUNÇÕES CONTÍNUAS.

Seja U um espaço de funções contínuas definidas sobre um intervalo $[t_0,t_f]$ .Uma entrada de controle, neste espaço é contínua em tal intervalo e tem a forma:

$$u = u(t) \quad t \in [t_0,t_f] \qquad (2.4.11)$$

Nosso objetivo é obter uma expressão para o gradiente do funcional de custo neste espaço. Logo na equação (2.4.10), devemos calcular $\frac{d\underline{x}}{d\varepsilon}(t_f,u)$. Integrando as equações do sistema (2.4.6), segue que:

$$\underline{x}(t_f,u) = \underline{x}(t_0) + \int_{t_0}^{t_f} f(\underline{x},u)\,d\tau \qquad (2.4.12)$$

Usando a equação (2.4.9) e derivando (2.4.12) com relação a ε:

$$\frac{d\underline{x}(t_f,u+\varepsilon z)}{d\varepsilon}\bigg|_{\varepsilon=o} = \frac{d\underline{x}(t_0)}{d\varepsilon}\bigg|_{\varepsilon=o} + \int_{t_0}^{t_f} f_x(\underline{x},u)\,\frac{d\underline{x}(t,u)}{d\varepsilon} + f_u(\underline{x},u)z(t)dt\bigg|_{\varepsilon=o} \tag{2.4.13}$$

onde $z = z(t)$, $t \in [t_0,t_f]$

e

$$(f_x(\underline{x},u))_{ij} = \frac{\partial f_i(\underline{x},u)}{\partial x_j} \qquad i,j = 1,2,3,\ldots,n \tag{2.4.14}$$

$$(f_u(\underline{x},u))_i = \frac{\partial f_i(\underline{x},u)}{\partial u} \qquad i = 1,2,3,\ldots,n \tag{2.4.15}$$

A equação (2.4.13), pode ser colocada em uma forma mais conveniente, eliminando-se alguns argumentos e voltando a usar a relação (2.4.9), obtem-se a expressão:

$$\frac{d\underline{x}(t_f,u)}{d\varepsilon} = \frac{d\underline{x}(t_0)}{d\varepsilon} + \int_{t_0}^{t_f} f_x\,\frac{d\underline{x}(t,u)}{d\varepsilon} + \underline{f}_u z(t)\,dt \tag{2.4.16}$$

Para o cálculo do gradiente é necessário resolver a equação integral acima. Isto é feito tornando-a mais genérica,considerando que a mesma deva ser válida para qualquer $t_f > t_0$. Desta forma, fazendo $t_f = t$ e diferenciando com relação a t, obtemos:

$$\frac{d}{dt}\frac{d\underline{x}(t,u)}{d\varepsilon} = f_x\,\frac{d\underline{x}(t,u)}{d\varepsilon} + f_u\,z(t) \tag{2.4.17}$$

fazendo: $\underline{X} = \dfrac{d\underline{x}(t,u)}{d\varepsilon}$ (2.4.18)

$$A(t) = f_x \tag{2.4.19}$$

$$\underline{B}(t) = \underline{f}_u \tag{2.4.20}$$

$$u = z(t) \ , \tag{2.4.21}$$

tem-se em (2.4.17) uma equação diferencial não homogênea de primeira ordem do tipo:

$$\dot{\underline{X}} = A(t)\underline{X} + \underline{B}(t)u \tag{2.4.22}$$

A solução desta equação é obtida, primeiro resolvendo-se a equação homogênea $\dot{\underline{X}} = A(t)\underline{X}$ e então introduzindo as variáveis de controle. Como solução desta equação tem-se (5,28,52):

$$\underline{X}(t) = \Phi(t,t_0)\underline{X}(t_0) + \int_{t_0}^{t} \Phi(t,\tau)\underline{B}(\tau)u(\tau)d\tau \tag{2.4.23}$$

onde $\Phi(t,t_0)$ é a matriz de transição(5), obtida da equação:

$$\frac{d\Phi(t,t_0)}{dt} = A(t)\Phi(t,t_0) \tag{2.4.24}$$

$$\Phi(t_0,t_0) = I \tag{2.4.25}$$

com I (5,28), sendo a matriz identidade (n X n) .

Substituindo as expressões (2.4.18) a (2.4.21) em (2.4.23):

$$\frac{d\underline{x}(t,u)}{d\varepsilon} = \Phi(t,t_0)\ \frac{d\underline{x}(t_0)}{d\varepsilon} + \int_{t_0}^{t} \Phi(t,\tau)\underline{f}_u z(\tau)d\tau \tag{2.4.26}$$

e recordando de (2.4.6) que $x(t_0) = c$, segue de (2.4.10):

$$\langle g(u),z\rangle = \left\langle \nabla_x \phi(\underline{x}(t_f,u)),\ \int_{t_0}^{t_f} \Phi(t_f,\tau)\ \underline{f}_u\ z(\tau)\ d\tau \right\rangle \tag{2.4.27}$$

usando que $\langle \underline{x}, B\underline{y}\rangle = \langle B^{\dagger}\underline{x}, \underline{y}\rangle$, com $B = \Phi(t_f,\tau)\underline{f}_u$, obtem-se:

$$\langle g(u), z\rangle = \int_{t_0}^{t_f} \left\langle \underline{f}_u^{\dagger} \phi^{\dagger}(t_f,\tau)\, \nabla_x \phi(\underline{x}(t_f,u)), z(\tau) \right\rangle \, d\tau \qquad (2.4.28)$$

Definindo $\underline{\lambda}(t) \equiv \phi^{\dagger}(t_f,t)\nabla_x\phi(\underline{x}(t_f,u))$ (2.4.29)

e notando que $z(\tau)$ é uma função escalar temos:

$$\langle g(u), z\rangle = \int_{t_0}^{t_f} \underline{f}_u^{\dagger}\underline{\lambda}(t) z(\tau)\, d\tau \qquad (2.4.30)$$

Da definição de produto escalar no espaço de funções contínuas e da equação acima, segue que o gradiente do funcional de custo , neste espaço é dado por:

$$\boxed{g(u(t)) = \underline{f}_u^{\dagger}\, \underline{\lambda}(t); \quad t \in [t_0, t_f]} \qquad (2.4.31)$$

Esta equação mostra, que para se calcular o gradiente é necessário determinar $\underline{\lambda}(t)$. Uma forma mais simples, que a equação (2.4.29) para se obter $\underline{\lambda}(t)$ é através da solução da equação adjunta:

$$\dot{\underline{\lambda}}(t) = -f_x^{\dagger}\underline{\lambda}(t); \qquad (2.4.32)$$

com $\lambda(t_f) = \nabla_x\phi(x(t_f,u))$

No apêndice A2, mostramos que $\underline{\lambda}(t)$, obtido de (2.4.29) satisfaz (2.4.32).

2.4.3 - O GRADIENTE NO ESPAÇO DE PARÂMETROS.

Neste caso, os elementos do espaço de entradas u em que se busca o mínimo $\in R^m$, onde $u_1, u_2, \ldots, u_m$ é o conjunto de parâmetros que se deseja otimizar.

A dinâmica do sistema a ser controlado também é descrita pela equação (2.4.6), com as considerações sobre a continuidade das derivadas parciais de até segunda ordem em $\underline{x}$ e $\underline{u}$. Em tal espaço, $\underline{u}$ tem a forma: $\underline{u} = [u_1, u_2, \ldots, u_m]$, e o funcional de custo $J[\underline{u}]$ que desejamos minimizar , também é descrito pela equação (2.4.7).

Podemos observar que se u_1 é um parametro de controle, ele pode ser considerado uma função contínua sobre o intervalo $[t_0, t_f]$ que é constante. Desta forma, as equações para a determinação do gradiente $\underline{g}[\underline{u}]$ do parágrafo (2.4.2) se aplicam aqui. Entretanto na variação do controle $\underline{u} + \varepsilon\underline{z}$, $\underline{z}$ agora tem a forma:

$$\underline{z} = [z_1, z_2, \ldots, z_m]^{\dagger} \tag{2.4.33}$$

onde z_i é uma constante sobre o intervalo $[t_0, t_f]$. Então da expressão (2.4.28), considerando que $\underline{z}$ é constante, podemos escrever:

$$\langle \underline{g}(\underline{u}), \underline{z} \rangle = \left\langle \int_{t_0}^{t_f} f_u^{\dagger} \phi^{\dagger}(t_f, \tau) \nabla_x \phi(\underline{x}(t_f, u)) d\tau, \underline{z} \right\rangle \tag{2.4.34}$$

e considerando $\underline{\lambda}(t)$ como dado por (2.4.29), temos:

$$\langle \underline{g}(\underline{u}), \underline{z} \rangle = \left\langle \int_{t_0}^{t_f} f_u^{\dagger} \underline{\lambda}(\tau) d\tau, \underline{z} \right\rangle \tag{2.4.35}$$

Da definição de produto escalar no R^m, o gradiente pode ser identificado como:

$$\boxed{\underline{g}(\underline{u}) = \int_{t_0}^{t_f} f_u^{\dagger} \underline{\lambda}(\tau) d\tau} \tag{2.4.36}$$

Cabe observar que, como agora $\underline{u}$ é um vetor m-dimensional, então

f_u é uma matriz (n X m) dada por:

$$f_u(x,y)_{ij} = \frac{\partial f_i(x,u)}{\partial u_j} \qquad \begin{matrix} i = 1,2, \ldots ,n \\ j = 1,2, \ldots ,m \end{matrix} \qquad (2.4.37)$$

2.4.4 - O GRADIENTE NO ESPAÇO DE ENTRADAS AMOSTRADAS.

Um caso de interesse prático em sistemas de controle é o caso do controle com entradas amostradas, como ilustrado na figura (2.4).

Vamos considerar que a entrada inicía em $t = t_0$ e muda de valor nos instantes de amostragem $t_0 + kT$, para $k = 0,1,2, \ldots ,N-1$, com $t_f = NT$. A entrada é considerada constante entre os intervalos de amostragem T, e pode ser expressa como função de t por:

$$u(t) = \sum_{k=0}^{N-1} u_k \left[\mathbb{1}(t-t_0-kT) - \mathbb{1}(t-t_0-(k+1)T) \right] \quad ,t \in [t_0,t_f] \qquad (2.4.38)$$

onde $\mathbb{1}(\cdot)$ é a função degrau unitário.

Para aplicar a variação $u + \varepsilon z$, z tem a forma:

$$z(t) = \sum_{k=0}^{N-1} z_k \left[\mathbb{1}(t-t_0-kT) - \mathbb{1}(t-t_0-(k+1)T) \right] \quad ,t \in [t_0,t_f] \qquad (2.4.39)$$

Como pode ser observado, u(t) e z(t) são completamente especificados por $u_1, u_2, \ldots , u_{N-1}$ e $z_1, z_2, \ldots , z_{N-1}$, podemos então considerar as entradas de controle como:

$$\underline{u} = [u_1,u_2, \ldots ,u_{N-1}]^{\dagger} \qquad (2.4.40)$$

com $\underline{u} \in R^N$.

Para obter o gradiente, observando u(t) em (2.4.38) pode-se notar que esta entrada é uma função contínua, com N-1 descontinuidades em $t_0 + kT$, para $k = 1,2,3,\ldots,N-1$, de forma que as equações do parágrafo 2.4.2, ainda se aplicam. Desta maneira, substituindo z(t) da equação (2.4.39) na equação (2.4.30) resulta:

$$\langle \underline{g}(\underline{u}), z \rangle = \int_{t_0}^{t_f} \underline{f}_u^{\dagger}\, \underline{\lambda}(t) \sum_{k=0}^{N-1} z_k \left[\mathbb{1}(t-t_0-kT) - \mathbb{1}(t-t_0-(k+1)T \right] dt$$

$$= \sum_{k=0}^{N-1} z_k \int_{t_0}^{t_f} \underline{f}_u^{\dagger}\, \underline{\lambda}(t) \left[\mathbb{1}(t-t_0-kT) - \mathbb{1}(t-t_0-(k+1)T \right] dt$$

$$= \sum_{k=0}^{N-1} \int_{t_0+kT}^{t_0+(k+1)T} \underline{f}_u^{\dagger}\, \underline{\lambda}(t)\, dt \cdot z_k \qquad (2.4.41)$$

E do produto escalar no R^N e da equação (2.4.41), o gradiente no espaço de entradas amostradas será:

$$\underline{g}(\underline{u}) = \left[g_0, g_1, g_2, \ \cdots\ , g_{N-1} \right]^{\dagger}$$

com g_k dado por

$$\boxed{g_k = \int_{t_0+kT}^{t_0+(k+1)T} \underline{f}_u^{\dagger}\, \underline{\lambda}(t)\, dt} \qquad (2.4.42)$$

onde $\underline{f}_u$ é obtida pela equação (2.4.15).

2.4.5- O GRADIENTE NO ESPAÇO DE MODULAÇÃO POR LARGURA DE PULSO.

caso A: Pulsos positivos e negativos

Vamos considerar que o sistema a ser controlado tenha sua dinâ-

mica descrita pela equação (2.4.6), com o funcional de custo dado por (2.4.7). Uma entrada de controle modulada por largura de pulso, definida sobre um intervalo de tempo $[t_0, t_f]$ pode ser equacionada como:

$$u(t) = \sum_{k=0}^{N-1} \operatorname{sgn}\tau_k \left[\mathbb{1}(t-t_0-kT) - \mathbb{1}(t-t_0-kT-|\tau_k|T) \right] \qquad (2.4.43)$$

onde sgn(·) é a função sinal, T é o intervalo de amostragem, N é o número de intervalos de amostragens, de forma que:

$$(t_f - t_0) = NT \qquad (2.4.44)$$

e $\tau_1, \tau_2, \ldots, \tau_{N-1}$ dão as larguras e sinais dos pulsos em t= 0, T, 2T, 3T,...,(N-1)T. A figura (2.5) representa um controle deste tipo.

Da equação (2.4.43), observa-se que u(t) fica completamente especificado por $\tau_1, \tau_2, \ldots, \tau_{N-1}$. Se consideramos um vetor $\underline{\tau} \in \Theta$ onde Θ é um subespaço de dimensão N do R^N dado por:

$$\Theta = \{ \underline{\tau}_i : |\tau_i| \leq 1;\ i=0,1,2,\ldots,N-1 \} \qquad (2.4.45)$$

e

$$\underline{\tau} = [\tau_1, \tau_2, \ldots, \tau_{N-1}] \qquad (2.4.46)$$

então para cada $\underline{\tau} \in \Theta$ corresponde um controle em modulação por largura de pulso dado por (2.4.43).

A integração de (2.4.6) com este dado controle, vai fornecer $x(t_f)$ que substituído em (2.4.7) dá $J[u]$ .Assim, para cada $\underline{\tau} \in \Theta$ existe $J[\underline{\tau}]$, e o problema pode ser redefinido para:determinar $\underline{\tau}$ que minimiza $J[\underline{\tau}]$ e o gradiente obtido é o de $J[\underline{\tau}]$.

O funcional de custo (2.4.7) pode ser reformulado em termos de

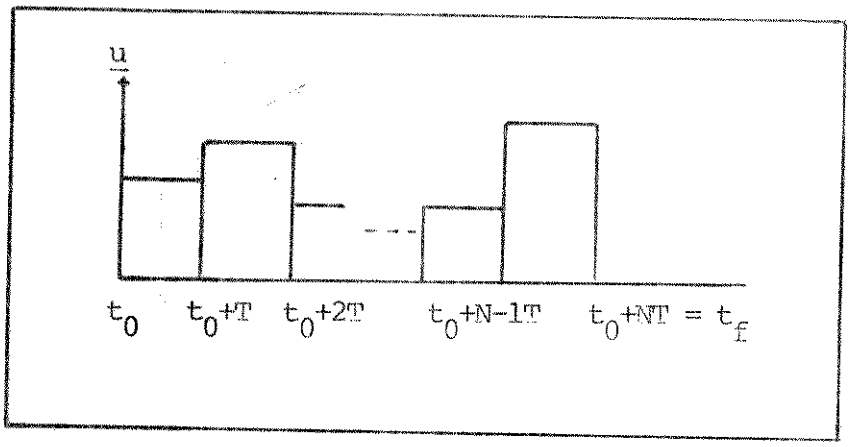


Fig. 2.4 - Representação de uma entrada de controle amostrada (28).

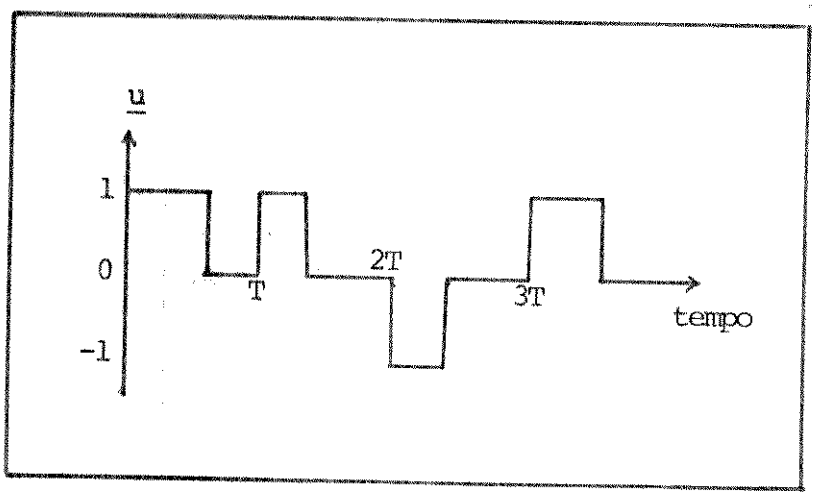


Fig. 2.5 - Representação de uma entrada de controle modulada por largura de pulso, para o caso A: pulsos positivos e negativos (28)

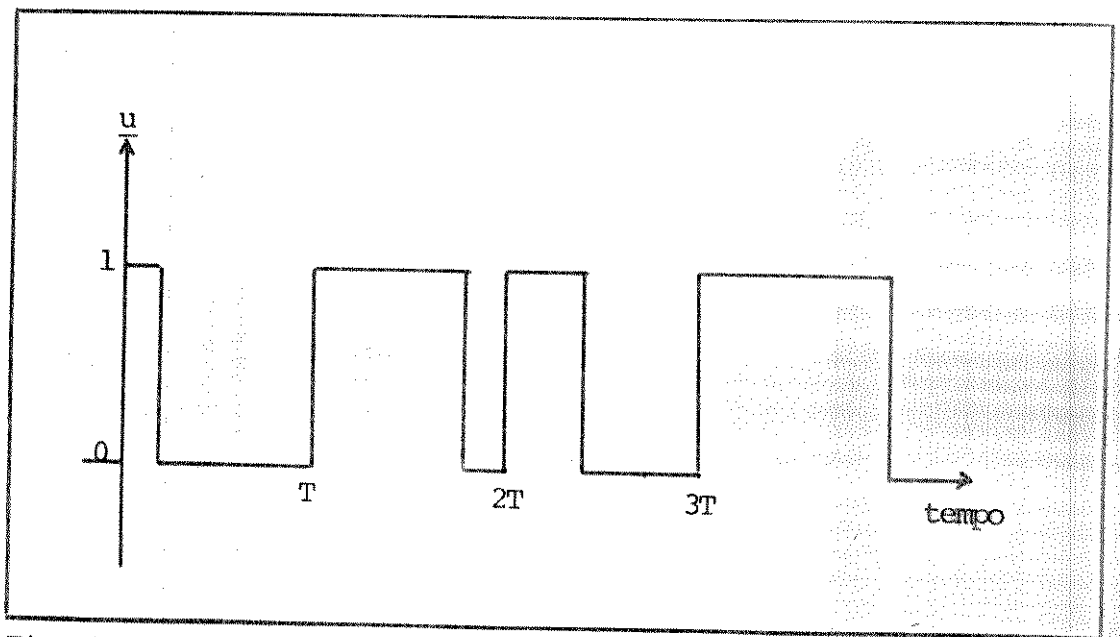


Fig. 2.6 - Representação de uma entrada de controle modulada por largura de pulso, para o caso B: pulsos positivos.

$\underline{\tau}$:

$$J[\underline{\tau}] = \phi(\underline{x}(t_f,\underline{\tau})) \qquad (2.4.47)$$

Utilizando (2.4.43), (2.4.6) pode ser integrada, resultando:

$$\underline{x}(t_f,\underline{\tau}) = \int_{t_0}^{t_f} \sum_{k=0}^{N-1} \{ \underline{f}(\underline{x},\mathrm{sgn}\tau_k) [\mathbb{1}(t-t_0-kT) - \mathbb{1}(t-T_k)] + \underline{f}(\underline{x},0) [\mathbb{1}(t-T_k) - \mathbb{1}(t-t_0-(k+1)T)] \} dt \qquad (2.4.48)$$

onde $T_k = t_0 + kT + |\tau_k| T$ é a borda de descida (ou de subida se τ_k é negativo) do k-ésimo pulso.

De modo a identificar o gradiente, devemos calcular $\frac{d\underline{x}}{d\underline{\varepsilon}}$ dado por (2.4.9) e substituir em (2.4.8). Usando $\underline{\tau} + \varepsilon\underline{z}$ em (2.4.48) obtemos:

$$\underline{x}(t_f, \underline{\tau} + \varepsilon\underline{z}) = c + \int_{t_0}^{t_f} \sum_{k=0}^{N-1} \{ \underline{f}(\underline{x},\mathrm{sgn}(\tau_k+\varepsilon z_k))\alpha + \underline{f}(\underline{x},0)\beta \} dt \qquad (2.4.49)$$

onde $\alpha = [\mathbb{1}(t-t_0-kT) - \mathbb{1}(t-t_0-kT-|\tau_k+\varepsilon z_k| T]$

$\beta = [\mathbb{1}(t-t_0-kT - |\tau_k+\varepsilon z_k| T) - \mathbb{1}(t-t_0-(k+1)T]$

Derivando (2.4.49) com relação a ε e calculando em $\varepsilon = 0$, resulta:

$$\frac{d\underline{x}(t_f,\underline{\tau})}{d\varepsilon} = \int_{t_0}^{t_f} \{ f_x \frac{d\underline{x}(t,\underline{\tau})}{d\varepsilon} + \sum_{k=0}^{N-1} [\underline{f}(\underline{x},\mathrm{sgn}\tau_k) - \underline{f}(\underline{x},0)] \cdot \delta(t-T_k)\mathrm{sgn}\tau_k \cdot T \cdot z_k \} dt \qquad (2.4.50)$$

$\frac{d\underline{x}}{d\underline{\varepsilon}}$ em (2.4.50) foi obtido de (2.4.9), f_x é dada por (2.4.14) e

$\delta(\cdot)$ é a função delta de Dirac.

A equação (2.4.50) é uma equação integral da mesma forma que a equação (2.4.16), cuja solução pode ser escrita de (2.4.26) como:

$$\frac{d\underline{x}(t_f,\underline{\tau})}{d\varepsilon} = \sum_{k=0}^{N-1} \Phi(t_f,T_k) \left[\underline{f}(\underline{x}(T_k),sgn\tau_k) - \underline{f}(\underline{x}(T_k),0)\right] sgn\tau_k . T . z_k \tag{2.4.51}$$

Φ é a matriz de transição de estado, com propriedades definidas em (2.4.24) e (2.4.25).

Substituindo a equação (2.4.51) em (2.4.8) resulta:

$$\langle \underline{g},\underline{z}\rangle = \left\langle \nabla_x \phi(\underline{x}(t_f,\tau)), \sum_{k=0}^{N-1} \Phi(t_f,T_k) \left[\underline{f}(\underline{x}(T_k),sgn\tau_k) - \underline{f}(\underline{x}(T_k),0)\right] sgn\tau_k . T . z_k \right\rangle \tag{2.4.52}$$

Usando (2.4.29) tem -se:

$$\langle \underline{g},\underline{z}\rangle = \sum_{k=0}^{N-1} \underline{\lambda}^{\dagger}(T_k) \left[\underline{f}(\underline{x}(T_k),sgn\tau_k) - \underline{f}(\underline{x}(T_k),0)\right] sgn\tau_k . T . z_k \tag{2.4.53}$$

Da equação (2.4.53) e da definição de produto escalar no R^N, a k-ésima componente do vetor gradiente é dada por:

$$g_k = \underline{\lambda}^{\dagger}(T_k) \left[\underline{f}(\underline{x}(T_k),sgn\tau_k) - \underline{f}(\underline{x}(T_k),0)\right] sgn\tau_k . T \qquad k = 1,2,\ldots,N-1 \tag{2.4.54}$$

Para calcular o gradiente pela equação (2.4.54), como no caso das funções contínuas, $\underline{\lambda}(t)$ pode ser mais fácilmente obtido por

(2.4.32) que por (2.4.29) .

caso B - Pulsos Positivos.

Um caso importante, e que não é tratado por Hasdorff(28), é quando os pulsos empregados para gerar as entradas moduladas possuem apenas uma polaridade (positiva ou negativa). Este caso ocorre mais frequentemente em situações práticas. Por exemplo, num sistema elétrico alimentado por uma tensão D.C., é mais fácil controlar a largura dos pulsos por meio de uma chave , ou circuito, liga - desliga, do que por um dispositivo, que além de ligar e desligar, em certas situações, deve comutar a polaridade da fonte.

Como outro exemplo, podemos citar o controle de um forno, alimentado por uma tensão A.C. de frequência w. Neste caso, se $w >> 2\Pi/T$, onde T é o intervalo de amostragem, a modulação por largura de pulsos pode ser efetuada, ligando-se e desligando a tensão de alimentação.

A dinâmica do sistema a ser controlado, bem como o funcional de custo são os mesmos do caso A. Entretanto a entrada de controle é descrita (veja figura 2.6) por:

$$u(t) = \sum_{k=0}^{N-1} \left[\mathbb{1}(t-t_0-kT) - \mathbb{1}(t-t_0-kT-|\tau_k|\ T) \right] \qquad (2.4.55)$$

para $t \in [t_0, t_f]$.

O subespaço do R^N, Θ é dado agora por:

$$\Theta = \{ \underline{\tau} : 0 \leq \tau_i \leq 1 \ , \ i = 0,1, \ldots, N-1 \} \qquad (2.4.56)$$

Feitas estas considerações e adotando o mesmo procedimento que no caso A, chegamos à seguinte expressão para o gradiente:

$$g_k = \lambda^{\dagger}(T_k)\left[\, \underline{f}(\underline{x}(T_k),1) - \underline{f}(\underline{x}(T_k),0) \,\right] \operatorname{sgn}\tau_k . T \qquad k = 0,1,2,\ldots,N-1 \qquad (2.4.57)$$

Na equação (2.4.55) foi considerado $|\tau_k|$, e mantido na dedução da equação(2.4.57), apenas por uma questão de segurança do ponto de vista computacional. Lembremos que, durante as iterações do método do gradiente conjugado, pode ocorrer de um dado elemento $\tau_k = 0^+$, se tornar 0^-, devido a algum ruído numérico. Neste caso a utilização de $|\tau_k|$ em (2.4.55) ajudaria a estabilizar o método.

2.5 - APLICAÇÃO DO MÉTODO DO GRADIENTE CONJUGADO À SOLUÇÃO DE PROBLEMAS DE CONTROLE ÓTIMO COM RESTRIÇÕES.

Muitos problemas de controle ótimo, particularmente aqueles de interêsse na engenharia, se apresentam com restrições, quer nas variáveis de estado, quer nas de controle. Tais restrições são provenientes das limitações físicas dos sistemas. Por exemplo, na obtenção do controle em tempo mínimo para um veículo percorrer determinada trajetória, a lei de controle está sujeita à capacidade de máxima aceleração e frenagem do veículo.

Problemas com restrições nas variáveis de estado e de controle, podem ser tratados pelas técnicas de *funções penalidades* (30,32,36, 37) e de *transformação* (13,52).

No primeiro método, de funções penalidades, o problema não é atacado diretamente, sendo reformulado como um problema irrestrito, onde termos de penalização não negativos, funções das variáveis vinculadas do problema original, são adicionadas ao critério de custo.

O termo de penalização é definido de forma a aumentar de valor, se a variável que lhe corresponde violar a restrição do problema inicial, desta forma elevando o valor do funcional de custo. Assim, segundo Fong(32), Lasdon(30) e Himmeblau(37), quando o novo funcional de custo for minimizado, as variáveis satisfarão as restrições do problema original.

Quanto às técnicas de transformação, Jacobson(53) foi um dos primeiros a propor uma técnica para a solução de problemas de controle ótimo, em que os vínculos de desigualdade nas variáveis de estado são transformados em vínculos de igualdade pela introdução de variáveis de folga. Derivadas temporais destes vínculos de igualdade associadas às equações de estado, formam um novo conjunto de equações. A expressão do controle nas equações de estado, e do funcional de custo em termos destas novas variáveis, geram um sistema sem restrições. Esta técnica é semelhante à empregada por Box (45) na solução de problemas de otimização estática. A principal desvantagem deste método está no fato do número de restrições ser menor ou igual ao número de equações de estado.

Ritch(13) desenvolveu uma técnica de transformação e separação dos vínculos para sistemas discretos, em que o número de restrições pode ser maior que o número de equações de estado. Neste método, as restrições sobre o estado são transformadas em restrições sobre o controle, e outra técnica deve ser empregada para tratar o problema, por exemplo o método de Pagurek descrito no próximo parágrafo. Segundo os autores(13,53), as técnicas de transformação são mais eficientes, e apresentam menos inconvenientes, que as de funções de penalidades.

Mais recentemente Miele (54) e colaboradores, desenvolveram uma nova técnica de transformação, que transforma restrições de

desigualdade, parcialmente lineares nas variáveis de estado, em restrições de igualdade. Entretanto este método não pode ser utilizado pelos algorítimos do gradiente conjugado, pois não elimina as restrições. Apesar disso este trabalho vem ilustrar o fato que a técnica de transformação ainda não está ultrapassada e continua sendo pesquisada.

2.5.1 - CONTROLE ÓTIMO COM RESTRIÇÕES NAS VARIÁVEIS DE CONTROLE: ALGORÍTMO DE PAGUREK.

Pagurek e Woodside(49) propuseram um algorítmo, que permite a utilização direta do método do gradiente conjugado na solução de problemas de controle ótimo, com restrições de desigualdade nas variáveis de controle.

Segundo os autores, se a variável u(t) tiver que satisfazer uma restrição de saturação da forma $|u(t)| \leq b$, quando da aplicação do método do gradiente, o simples fato de estabelecer $u_{i+1}(t) = \pm b$, conforme seja o caso, leva a um valor do funcional de custo maior que o ótimo. É proposto que, se o controle ótimo u*(t) saturar em uma região W do intervalo $[t_o, t_f]$ e se W for conhecido a priori, $u_i(t)$ para $t \in W$, pode ser feito igual a ±b, conforme o caso. Devido a falta de liberdade em se escolher u_{i+1} para $t \in W$, $p_i(t)$ e $g_i(t)$ devem ser igualados à zero neste intervalo (49), e a busca do ótimo deve ser concentrada fora da região W. Consequentemente, para $t \in W$ obtemos:

$$u_{i+1}(t) = u_i(t) = u^*(t) \quad e \quad p_i(t) = 0 \qquad (2.5.1)$$

Se W não é conhecido a priori, então o intervalo de saturação para $u_{i+1}(t)$ é considerado o mesmo que o de $u_i(t)$. Desta forma, inician

do com $u_i(t)$ e p_i para $t \in [t_0, t_f]$, considera-se que W_i é a região de saturação de $u_i(t)$. Definindo a função escalar $w_i(t)$ tal que:

$$w_i = \begin{cases} 0 \text{ para } t \in W_i \\ 1, \text{em qualquer outra parte,} \end{cases} \qquad (2.5.2)$$

procede-se como no caso sem restrições, somente que no cálculo de β_i usa-se $w_i(t)g_{i+1}(t)$, em lugar de $g_{i+1}(t)$.

De forma a dar condições, para que o intervalo W_i se altere, calcula-se:

$$u_{i+1}(t) = u_i(t) + \alpha_i p_i(t) \qquad (2.5.3)$$

e

$$p_{i+1}(t) = -g_{i+1}(t) + \beta_i p_i(t) \qquad (2.5.4)$$

Durante a busca unidimensional, o cálculo do custo $J[u_{i+1}(t)]$ deve ser efetuado com $u_{i+1}(t)$ truncado nos limites superior e inferior, quando houver saturação. Consequentemente o resultado da busca unidimensional é uma trajetória nova e melhor (49), que satura no intervalo $W_{i+1} \neq W_i$.

CAPITULO 3

IMPLEMENTAÇÃO DO ALGORÍTMO DE MINIMIZAÇÃO PELO MÉTODO DO GRADIENTE CONJUGADO: GRAMIN.

No presente capítulo desenvolvemos um programa geral chamado GRAMIN, para obtenção de entradas de controle, que minimizam um dado critério de custo durante a operação do sistema.

O programa faz uso do método do Gradiente Conjugado, e dos procedimentos anteriormente descritos para obtenção do gradiente nos seguintes espaços de controle: funções contínuas, conjunto de parâmetros, entradas amostradas e modulação por largura de pulso. Além destes quatro espaços, o programa pode se estender a um espaço genérico definido pelo próprio usuário. Outra característica do GRAMIN é a capacidade de manipular restrições de saturação nas variáveis de controle, através do algorítmo de Pagurek(49).

Na sequência deste capítulo, além de tratar da implementação do GRAMIN, daremos instruções para o uso do programa.

3.1- IMPLEMENTAÇÃO DO GRAMIN.

Para a implementação do GRAMIN é necessário recordarmos os passos básicos envolvidos no algorítmo do Gradiente Conjugado, descritos no capítulo anterior através das equações (2.3.15) a (2.3.19) e aplicados ao espaço de controle de interêsse.

A dinâmica do sistema, para o qual queremos obter as entradas de controle , é dada pela equação (2.4.6), e o funcional de custo para uma dada entrada de controle $\underline{u}$ é dado pela equação (2.4.7),

que transcrevemos abaixo:

$$\frac{d\underline{x}(t)}{dt} = \dot{\underline{x}}(t) = \underline{f}(\underline{x}(t),\underline{u}(t)) \; ; \; \underline{x}(t_0) = C \qquad (3.1.1)$$

$$J = J[\underline{u}] = \phi(\underline{x}(t_f,\underline{u})) \qquad (3.1.2)$$

Para cada espaço de controle considerado, o funcional de custo (equação 3.1.2) pode ser calculado a partir de seu argumento $\underline{u}$.

Da mesma forma o gradiente do funcional $\underline{g}(J(\underline{u}))$ pode ser obtido, através dos procedimentos apresentados na seção 2.4.0 problema básico resolvido pelo GRAMIN, é obter um $\underline{u}^*$, tal que $J(\underline{u})$ é um mínimo.

O programa localiza $\underline{u}^*$, iterativamente através de uma sequência $\underline{u}_0, \underline{u}_1, \ldots, \underline{u}_n$ onde $\underline{u}_0$ é uma estimativa inicial e $J(\underline{u}_{i+1}) < J(\underline{u}_i)$. O limite dessa sequência é o valor $\underline{u}^*$ procurado. Entretanto, no GRAMIN para limitar o tempo de computação, o número de iterações na sequência, é definido pelo usuário.

3.1.1 - IMPLEMENTAÇÃO DO SUBPROGRAMA GRAMI.

Com as considerações do parágrafo anterior, o algorítmo do Gradiente Conjugado, é desenvolvido nas seguintes etapas:

1- Partindo de uma estimativa $\underline{u}_0$, calcula-se o gradiente $\underline{g}(J(\underline{u}_0)) = \underline{g}_0$ para o espaço de controle em questão, e toma-se um vetor $\underline{p}_0$, tal que $\underline{p}_0 = -\underline{g}_0$, equação (2.3.15);

Para i = 0,1, ... n:

2- Determina-se por meio de uma busca unidimensional (apêndice

A.1), o escalar $\alpha_i > 0$ que minimiza $J(\underline{u}_i+\alpha_i\underline{p}_i)$, equação (2.3.17).

3- Calcula-se $\underline{u}_{i+1} = \underline{u}_i+\alpha_i\underline{p}_i$, equação (2.3.16).

4- Obtem-se novo $\underline{p}_i$ tal que $\underline{p}_{i+1} = -\underline{g}_{i+1} + \beta_i\underline{p}_i$ equação (2.3.18) onde $\beta_i = \dfrac{\langle \underline{g}_{i+1}, \underline{g}_{i+1} \rangle}{\langle \underline{g}_i, \underline{g}_i \rangle}$, equação (2.3.19) e $g_{i+1} = \underline{g}[J(\underline{u}_{i+1})]$.

Com este algorítmo construímos o subprograma GRAMI, porém como o próprio algorítmo mostra, é necessário a construção de alguns subprogramas auxiliares:

a - GRAD, calcula o gradiente para cada espaço de entradas de controle, segundo a teoria desenvolvida na seção 2.4.

b - BUSCA, realiza a busca unidimensional para encontrar α_i.

c - ESCSU, determina o produto escalar de dois elementos, considerando o espaço em que se está trabalhando.

Além destes subprogramas que são imediatos do algorítmo apresentado, outros se fazem necessários para a inicialização de constantes e dimensões, e para implementação do algorítmo de Pagurek. São eles:

d - INITI, realiza a inicialização das constantes do Runge - Kutta, e dimensões comuns aos subprogramas.

e - PAGI, é utilizado quando se quer adicionar o método de Pagurek, para o tratamento das variáveis de controle com restrições de saturação.

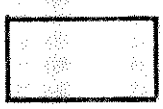
Na figura 3.1, temos o fluxograma simplificado do subprograma GRAMI. O GRAMI, juntamente com todos os outros subprogramas auxiliares constitui o "pacote computacional" chamado programa GRAMIN.

3.1.2 - PROCEDIMENTO PARA O CÁLCULO DO GRADIENTE NOS ESPAÇOS DE ENTRADAS DE CONTROLE DE INTERÊSSE.

Os espaços de entradas de controle considerados são:

LEGENDA DA FIGURA 3.1 :

Este bloco contem elementos que são sub-rotinas.



Bloco de atribuição.

Bloco de decisão.

1 - As variáveis não relevantes para o entendimento do algorítmo como um todo, foram suprimidas.

2 - O mesmo foi feito com os contadores de índices, para operações vetoriais.

3 - Afirmação do tipo X(i) ← A , significa que o valor de A é atribuído a todas as componentes de X(i).

4 - Em subprogramas, os argumentos grifados representam valores de saída e os demais de entrada.

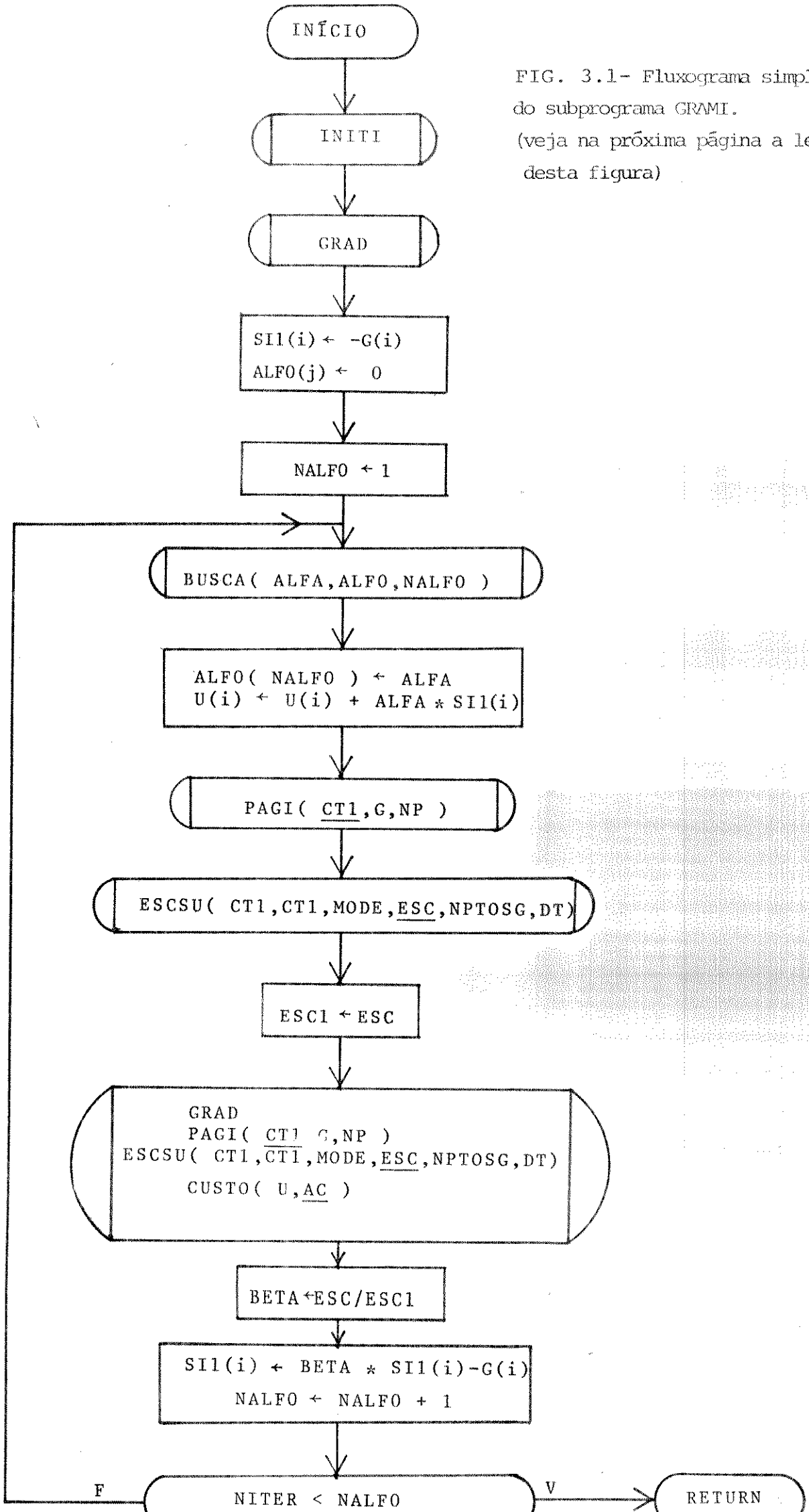


FIG. 3.1- Fluxograma simplificado do subprograma GRAMI.
(veja na próxima página a legenda desta figura)

a) funções contínuas;

b) espaço de parâmetros;

c) entradas amostradas;

d) modulação por largura de pulso;

e) espaço do usuário.

Para os espaços dos ítens (a) até (d), a dinâmica do sistema, o funcional de custo, bem como as expressões analíticas para o gradiente, já foram definidas e obtidas na seção 2.4 do capítulo anterior. No caso de espaço do usuário, a dinâmica do sistema e o funcional de custo são definidos da mesma forma, porém o usuário seguindo um procedimento análogo ao da seção 2.4, obtem uma expressão analítica para o gradiente no espaço de seu interêsse.

Com o objetivo de implementar o programa GRAD, resumiremos os passos necessários para a obtenção do gradiente, em cada espaço considerado.

a) Espaço de Funções Contínuas.

Neste espaço a entrada de controle u(t) é uma função contínua definida no intervalo $[t_0, t_f]$, equação (2.4.11), e a expressão do gradiente é dada pela equação (2.4.31):

$$g(u(t)) = \underline{f}_{\underline{u}}^{\dagger}\underline{\lambda}(t) \quad ; \quad t \in [t_0, t_f] \qquad (3.1.3)$$

Para calcular g(u(t)), deve-se executar os seguintes passos:

1 - integrar de t_0 a t_f as equações do sistema (ver equação 3.1.1), usando a entrada u(t) dada , para obter o estado do sistema x(t) neste intervalo de tempo.

2 - obtido $\underline{x}(t_f)$, integra-se a equação (2.4.32) transcrita abai

xo, de t_f até t_0 para obter $\underline{\lambda}(t)$ em tal intervalo.

$$\dot{\underline{\lambda}}(t) = -f_x^{\dagger}\,\underline{\lambda}(t) \quad ; \qquad (3.1.4\text{-a})$$

$$\underline{\lambda}(t_f) = \nabla_x\,\phi(x(t_f,u)) \quad ; \qquad (3.1.4\text{-b})$$

3 - através da expressão (3.1.3) calcula-se o gradiente $g(u(t))$.

b) Espaço de Parâmetros.

Aqui a entrada de controle u, é um conjunto de parâmetros do tipo: $\underline{u} = [u_1, u_2, \ldots, u_m]$ onde $\underline{u} \in R^m$.

O gradiente pode ser agora encontrado, repetindo-so os passos 1 e 2 do caso (a), para a obtenção de $\lambda(t)$ no intervalo $[t_0,t_f]$, e pela aplicação da equação (2.4.36):

$$\underline{g}(\underline{u}) = \int_{t_0}^{t_f} f_{\underline{u}}^{\dagger}\underline{\lambda}(\tau)\,d\tau \qquad (3.1.5)$$

c) Espaço de Entradas Amostradas.

Na seção 2.4.4 foi estudado o caso de entradas de controle amostradas, com u(t) dado pela equação (2.4.38) e cuja forma está ilustrada na figura (2.4).

Para a obtenção do gradiente neste espaço de entradas de controle, determina-se $\lambda(t)$ como nos passos 1 e 2 do caso de funções contínuas e então se faz uso da equação (2.4.42):

$$g_k = \int_{t_0+kT}^{t_0+(k+1)T} \underline{f}_{\underline{u}}^{\dagger}\underline{\lambda}(t)\,dt \qquad (3.1.6)$$

para a obtenção da k-ésima componente do gradiente, onde k = 0,1,..., N-1 .

d) Entradas de Controle Moduladas por Largura de Pulso.

As entradas de controle neste espaço estão esquematizadas nas figuras 2.5 (pulsos positivos e negativos) e 2.6 (pulsos positivos). A determinação do gradiente é feita repetindo-se os passos 1 e 2 do caso de funções contínuas e utilizando-se das equações para a k-ésima componente do gradiente, equação (2.4.54) para o caso de pulsos positivos e negativos e (2.4.57) para pulsos positivos.

e) Espaço do usuário.

Caso o usuário necessite trabalhar em um espaço de entradas de controle diferente dos tratados até então, ele poderá por um procedimento semelhante ao adotado na seção 2.4, obter sua própria expressão para o gradiente. Nesta expressão o gradiente aparecerá como uma função de $\lambda(t)$ e $f_u(t)$ como nos espaços anteriores. E novamente os passos 1 e 2 do caso de funções contínuas devem ser repetidos, para a determinação de $\lambda(t)$.

Pode-se concluir então, que para qualquer espaço de entradas de controle considerado, eles têm em comum a determinação de $\lambda(t)$, diferindo apenas na expressão que fornece o gradiente.

3.1.3 - IMPLEMENTAÇÃO DO SUBPROGRAMA GRAD.

O subprograma GRAD, calcula o gradiente de um dado funcional de custo, em qualquer dos espaços de controle considerados neste trabalho. Todos os procedimentos para o cálculo do gradiente, em cada espaço de controle, apresentados na seção 3.1.2, podem ser divididos em quatro etapas:

1-integrar a equação de estado do sistema de t_0 até t_f para obter o estado final do sistema $x(t_f)$;

2-obter o estado final da equação adjunta do sistema, equação (3.1.4-b);

3-integrar a equação adjunta de t_f até t_0, para obter $\lambda(t)$, para qualquer instante t;

4-determinação do gradiente, de acordo com o espaço considerado, partindo de uma função de $f_u(t)$ e $\lambda(t)$.

Para a implementação destes 4 passos, o subprograma chama as seguintes sub-rotinas:

a-RUNGE, que por sua vez usa a sub-rotina FUNX para resolver a equação de estado, e a sub-rotina FUNL, para resolver a equação adjunta do sistema;

b-GRAXF, sub-rotina para a obtenção da condição terminal $\lambda(t_f)$ da equação adjunta (equação 3.1.4-b);

c-GRADI , que fornece a relação entre $f_u(t)$ e $\lambda(t)$ para o cálculo do gradiente em cada espaço. Quem seleciona cada um dos cinco espaços de trabalho, é o subprograma GRAD, através de uma variável MODO setada pelo usuário.

d-SYMP , como em alguns espaços é necessário fazer a integração da saída da sub-rotina GRADI, a sub-rotina SYMP realiza tal função.

Alem disso, o subprograma GRAD chama uma sub-rotina USUB , que não está explícita no algorítmo acima (passos 1 a 4), mas que é usada no caso de modulação por largura de pulso, para gerar a entrada de controle.

3.1.4 - IMPLEMENTAÇÃO DA BUSCA.

O subprograma BUSCA determina o escalar $\alpha_i > 0$ que, a cada ite

ração i, minimiza a equação (2.3.17). O algorítmo desenvolvido utiliza uma combinação do método de Davies-Swann-Campey (DSC) e o de Powell (vide apêndice A1), com estimativa do passo inicial.

De todos os algorítmos tratados aqui, o da BUSCA é o mais crítico, uma vez que as propriedades de convergência e velocidade do GRAMI dependem diretamente dele. Isto ocorre porque cada passo da Busca Unidimensional envolve o cálculo do funcional de custo, que pela equação (2.4.7) é uma função de $\underline{x}(t_f,u)$, que por sua vez deve ser obtido por integração da equação (2.4.12) de t_0 até t_f. A necessidade de se reduzir o número de cálculos da função objetivo, impõe que na fase onde é usado o algorítmo de Powell (37), a interpolação quadrática seja realizada apenas uma vêz. Alguns autores tais como Hasdorff (28) e Fletcher & Reeves (44), tentam compensar este fato, realizando uma interpolação cúbica. Entretanto na própria referência (44), os autores afirmam que não se consegue uma redução significante no número de iterações, usando interpolação de ordem maior que 3. Além do mais a interpolação cúbica usada na referência (44) tem a desvantagem de necessitar de informações sobre o gradiente nos pontos de interpolação. Por este motivo, neste trabalho será utilizado o algorítmo de Powell que requer apenas três valores do funcional de custo e de seu argumento para realizar a interpolação (37).

O Algorítmo da BUSCA.

Cada vez que é chamado, o subprograma BUSCA usa um vetor ALF0 e um escalar NALF0 como parâmetros de entrada, e como saída, um escalar ALFA. Na i-ésima iteração do GRAMI, o vetor ALF0 contém os resultados das buscas unidimensionais anteriores (α_i, para i=0,1,...,i-1), e NALF0 fornece o número de elementos nesse vetor. Terminado o subprograma BUSCA, ALFA contém o valor de α_i procurado. O algorítmo da BUSCA, em sua

forma mais completa, compreende os passos descritos abaixo.

1 - Esta é a primeira vêz que a BUSCA é chamada?

Caso afirmativo, segue para o passo 2, se a resposta for negativa continua no passo 4.

2 - Estimativa do passo inicial.

A estimativa do passo inicial Δx é feita considerando uma expansão em série de Taylor do funcional $F(.)$ no ponto x_0, com termos de primeira ordem:

$$F(\underline{x}_0 + \Delta x \underline{p}_0) = F(\underline{x}_0) - \Delta x \langle \underline{g}_0, \underline{g}_0 \rangle \qquad (3.1.7)$$

Para a expansão ser verdadeira, Δx deve ser pequeno. Mas se Δx é pequeno, $F(\underline{x}_0 + \Delta x \underline{p}_0)$ não deve diferir muito de $F(\underline{x}_0)$. Um valor arbitrário, mas razoável é considerar:

$$F(\underline{x}_0 + \Delta x \underline{p}_0) - F(\underline{x}_0) \simeq -0{,}03F(\underline{x}_0) \qquad (3.1.8)$$

Destas equações (3.1.7) e (3.1.8) Δx pode ser obtido:

$$\Delta x = \frac{0.03F(\underline{x}_0)}{\langle \underline{g}_0, \underline{g}_0 \rangle} \qquad (3.1.9)$$

3- Estabilização da busca.

Para estabilizar a busca, Δx é dividido por 2^{10}. Segue para o passo 6.

4- O passo inicial é estimado como média geométrica dos escalares α_i para i=0,1,...,i-1. Sendo que estes valores se encontram disponíveis como elementos de ALF0.

$$\Delta x = (\alpha_0 * \alpha_1 * \ldots * \alpha_{i-1})^{1/i} \qquad (3.1.10)$$

5- De forma a manter a estabilidade da busca, Δx é dividido por 2^3.

6- Busca da região em que o mínimo está localizado (bracket):

Para simplificar a notação seja:

$$F(\Delta \underline{x}_i) = F(\underline{x}_i + \Delta x \underline{p}_i) \qquad (3.1.11)$$

e nesta notação $F(0) = F(\underline{x}_i)$. (3.1.12)

Para k = 1,2,3,... , calcula-se:

$$\Delta x_k = 2^k \Delta x_i \quad \text{se} \quad F(\Delta x_i) < f(0) \quad ; \qquad (3.1.13)$$

ou

$$\Delta x_k = 2^{-k} \Delta x_i \quad \text{se} \quad F(\Delta x_i) \geq F(0) \quad . \qquad (3.1.14)$$

$$\text{Se} \quad F(\Delta x_k) > F(\Delta x_{k-1}) \qquad (3.1.15)$$

o processo é interrompido e o mínimo procurado, está localizado entre os pontos $\Delta x_k = x_3$, $\Delta x_{k-1} = x_2$, $\Delta x_{k-2} = x_1$.

7- Estreitando o intervalo em que o mínimo se localiza.

Os pontos D e E são determinados, de forma que D é o ponto médio entre x_1 e x_2 e E é o ponto médio entre x_2 e x_3 Dentre estes cinco pontos, determina-se qual dos três definem o intervalo em que se localiza o mínimo, sendo que os outros dois são descartados. Estes três pontos são redefinidos como x_3 , x_2 , x_1 e o processo é repetido por quatro vezes antes de encaminhar ao passo 8.

8- Realiza a interpolação quadrática, pela equação de Powell (vide apêndice A1), para a obtenção do ponto $\tilde{x}^*$.

9- Compara-se $F(\tilde{x}^*)$ e $F(x_2)$

Se $F(\tilde{x}^*) < F(x_2)$ então toma-se ALFA = $\tilde{x}^*$.

Se $F(\tilde{x}^*) \geq F(x_2)$ então toma-se ALFA = x_2.

10- NALF0 é incrementado e retorna-se ao subprograma GRAMI.

Comentários sobre a BUSCA.

O algorítmo da Busca Unidimensional implementado no subprograma BUSCA, e descrito nos passos 1 a 10 , está na forma chamada comple

ta. Nesta forma, foram acrescentados os passos 3,5, e 7 que, apesar de tornar o algorítmo mais lento, aumentam sua precisão. A Busca na forma incompleta (b) não executa os passos 3 e 5, e na forma incompleta (a) estes três passos adicionais não existem. Quanto ao tempo de computação, a busca mais lenta é a completa, e a mais veloz é a incompleta (a), sendo que esta última requer 50% do tempo de execução da completa.

Os passos 3 e 5, tendem a levar o primeiro ponto calculado pela Busca , mais próximo do ponto inicial. Este procedimento é indicado, principalmente quando o funcional apresentar uma alternância entre regiões de valor constante e regiões de variações súbitas em seu valor, como ocorre quando há saturação na variável de controle. Neste caso uma estimativa de primeira ordem do mínimo não é suficiente e é desejável que os pontos iniciais da busca estejam próximos de x_0, para evitar que a Busca possa convergir para um outro mínimo local mais afastado de x_0.

A finalidade do passo 5 é melhorar a precisão com que o mínimo é interpolado, pois quanto mais próximos estiverem os três pontos, menor será o erro na interpolação quadrática. A precisão na determinação de α é importante, pois a dedução do método de minimização pelo gradiente conjugado, está baseada no fato de se poder calcular o mínimo de $F(\underline{x}_i + \alpha_i \underline{p}_i)$.

3.1.5 - OS SUBPROGRAMAS PAGI E PAG.

O subprograma PAGI implementa o algorítmo de Pagurek, que é usado nos casos em que há restrições de saturação nas variáveis de controle (veja seção 2.5.1). O PAGI é chamado no programa GRA-

MI, antes de ser realizado o produto escalar $\langle \underline{g}_{i+1}, \underline{g}_{i+1} \rangle$, para verificar se há saturação em algum ponto da variável de controle $u_i(t)$. Havendo saturação em um intervalo $W_i \in [t_0, t_f]$, o vetor $\underline{g}_{i+1}(t)$ para $t \in W_i$ é igualado a zero, e as outras componentes $\underline{g}_{i+1}(t)$, tal que $t \notin W_i$ são mantidas. Contudo o vetor original $\underline{g}_{i+1}(t)$ é preservado para cálculos futuros. Se não houver saturação o subprograma PAGI simplesmente mantem o vetor original.

O uso do algorítmo de Pagurek, é uma opção do usuário, sendo definida por uma variável lógica IPGK. No início do PAGI é feito um teste sobre a condição de IPGK. Se seu valor for zero, retorna-se ao programa principal havendo ou não saturação. Com IPGK = 1, é feito um teste para verificar se a chave MODO é igual a 5 (espaço do usuário). Em caso afirmativo é chamado o subprograma PAG. Este subprograma tem funções idênticas ao PAGI, porém é definido pelo usuário, para atender as restrições de saturação em seu espaço.

3.1.6 - OS SUBPROGRAMAS ESCSU, SYMP, INITI E RUNGE.

O subprograma ESCSU realiza o produto escalar de dois elementos no espaço de entradas de controle que se está trabalhando. Nestes espaços, a operação de produto escalar se resume em 2 tipos: produto escalar de funções, que requer integração, e produto escalar de vetores que envolve somatórias do produto de suas componentes. Existe neste subprograma uma variável MODE que seleciona o tipo de operação de produto escalar. Se MODE = 0 , tem-se o produto escalar de vetores. Se MODE = 1 , trata-se do produto escalar de funções e neste caso é chamada a sub-rotina SYMP.

A sub-rotina SYMP, realiza a integração numérica, através da regra de Simpson (56), cujo êrro de integração é da ordem de h^4, onde h é o intervalo de discretização da função.

Na sub-rotina INITI, são calculadas as constantes comuns a vários

subprogramas. Como a INITI é executada apenas uma vez, sua função é evitar que tais constantes sejam calculadas redundantemente.Um exemplo mais significativo, é o cálculo dos coeficientes de Runge, que se estivessem na sub-rotina RUNGE, seriam calculados cada vez que a mesma é chamada. Pode-se avaliar tal economia considerando que,com discretizações de 200 pontos e para 5 iterações do Gradiente Conjugado, a RUNGE é chamada cerca de 1500 vezes, no caso da busca completa .

O subprograma RUNGE resolve a equação de estado do sistema , partindo da condição inicial $x(t_0)$, e a equação adjunta, partindo da condição terminal $x(t_f)$. Existe no subprograma RUNGE, uma variável lógica XOUL que define qual das duas equações será resolvida. Quando XOUL = 0, o RUNGE chama uma sub-rotina do usuário FUNX, que define a equação de estado do sistema. Se XOUL = 1, o subprograma chama a sub-rotina FUNL, também fornecida pelo usuário e que define a equação adjunta. Para realizar a integração da equação de estado ou da adjunta, o RUNGE é chamado n vezes onde n é o número de intervalos de discretização entre t_0 e t_f. Este subprograma foi implementado com o algorítmo de Runge-Kutta de quarta ordem, usando o método de Gill (21).

3.2 - INSTRUÇÕES PARA O USO DO GRAMIN.

Um usuário que tenha um sistema definido por uma equação do tipo (2.4.6), com custo de operação na forma da equação (2.4.7) e que deseje encontrar um controle $\underline{u}$ ótimo para minimizar seu funcional de custo, deve proceder da seguinte maneira:

1-Obter a expressão para o gradiente utilizando os procedimentos da seção 2.4 .

2-Escrever a equação adjunta para o sistema (2.4.32) com sua condição terminal.

3-Construir um Programa Principal, onde são definidas as constantes de inicialização do GRAMIN e que chame o programa GRAMI como sub-rotina.

4-Escrever as seguintes sub-rotinas que serão chamadas pelo GRAMIN: FUNX, FUNL, GRAXF, GRADI, FI e opcionalmente PAG .

5-Definir no Programa Principal e em cada uma das sub-rotinas acima as linhas com a declaração COMMON, nesta ordem:

```
COMMON U(201),G(201),X(10,201),ALBDA(10,201)
COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP
COMMON XO(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
COMMON A(4),B(4),C(4),N11
```

3.2.1 - PROGRAMA PRINCIPAL.

No Programa Principal deve-se definir as seguintes constantes de inicialização do GRAMIN:

MODE - determina o tipo de produto escalar a ser empregado.

MODE = 0 para produto escalar de vetores.

MODE = 1 para produto escalar de funções.

MODO - define o espaço de entradas de controle.

MODO = 1 , espaço de funções contínuas.

MODO = 2 , espaço de parâmetros.

MODO = 3 , entradas amostradas.

MODO = 4 , espaço de modulação por largura de pulso (MLP)

MODO = 5 , espaço do usuário.

N - indica a dimensão do vetor de estado.

M - fornece a dimensão da entrada de controle.

TI - define o instante de tempo inicial.

TF - idem para o instante de tempo final.

NPTOS - dá o número de intervalos de discretização do vetor de estado, entre t_0 e t_f.

NPTOSG - determina o número de intervalos de discretização do gradiente, entre t_0 e t_f.

Estas duas últimas constantes devem ser números pares, pelo fato do GRAMIN fazer uso da regra de Simpson.

NITEP - define o número de iterações a serem realizadas pelo Gradiente Conjugado.

X0 - vetor de N componentes definindo o estado inicial do sistema.

U - vetor de M componentes definindo a estimativa inicial do controle ótimo.

Após a definição destas constantes, se o Programa Principal chamar uma vez a sub-rotina GRAMI, quando esta terminar a sua execução o vetor U contem o controle ótimo após as iterações definidas em NITER, e a componente X_{ij}(i = 1,2,...,N e j = 1,2,...,NPTOS+1) da matriz X definida no COMMON, contem a i-ésima componente do vetor de estado $x(t_j)$, e $x(t_{j=NPTOS}) = x(t_f)$. Portanto se o usuário quizer reinicializar o método, partindo do controle e do estado assim encontrado, ele deve chamar novamente o subprograma GRAMI. Este prodimento no capítulo 2 foi chamado reinicialização do método. É possível chamar o subprograma GRAMI, dentro de um loop de contagem para fazer sucessivas reinicializações.

OBSERVAÇÕES:

1- As dimensões máximas usadas no GRAMIN, foram iguais a 10

para o número de componentes do vetor de estado e 201 para o número de pontos de discretização. As demais variáveis acompanham tal dimensionamento, dependendo do tipo de variável utilizada.

2 - Se o usuário definir alguma declaração COMMON para ser usada em seu subprograma, esta declaração deve suceder as linhas de COMMON já estabelecidas.

3.2.2 - SUB-ROTINAS.

FUNX - Subprograma que define a equação de estado do sistema. A sub-rotina FUNX deve ser definida pela seguinte declaração,

```
SUBROUTINE FUNX(T,H,F,UDT)
```

e conter as linhas:

```
DIMENSION UDT(201),F(11)
I = (T-TI)/DT+1
```

onde UDT(I) fornece o valor de u(t) nos intervalos de discretização e F(J) deve ser definida pelo usuário como a j-ésima componente da equação de estado no instante I, ou seja $F(J) = \dot{x}_i(I)$.

FUNL - Subprograma que define a equação adjunta. Os seguintes argumentos de FUNX devem ser definidos em FUNL,

```
SUBROUTINE FUNL(T,H,F,UDT)
```

e também as linhas:

```
DIMENSION UDT(201),F(11)
I = (T - TI)/DT+1
```

com UDT(I) definido como em FUNX, porém agora F(J) fornece a j-ésima componente da equação adjunta no instante I, ou $F(J) = \dot{\lambda}_i(I)$.

GRAXF - Subprograma para calcular a condição terminal $\lambda(t_f)$. A GRAXF não requer argumentos sem efeito em sua declaração, e nela a condição terminal $\lambda(t_f)$ deve ser calculada (equação 2.4.32) e o resultado atribuído à variável Xl(I).

GRADI - Sub-rotina que define a relação entre f_u e $\lambda(t)$ no cálculo do gradiente. O subprograma GRAD faz uso da relação entre f_u e $\lambda(t)$ definida nesta sub-rotina, para obtenção do gradiente. Ela é declarada como:

```
SUBROUTINE GRADI(GA,I,J)
```

onde GA é um argumento de saída e fornece o valor da relação entre f_u e $\lambda(t)$ como função dos argumentos de entrada I e J. As variáveis I e J têm sua utilização variada, segundo o espaço de entradas de controle considerado. Por exemplo, no espaço de funções contínuas, I representa o intervalo de discretização em que se está calculando o gradiente e J não é utilizado. O uso das variáveis I e J, na sub-rotina GRADI, será melhor esclarecido nas listagens dos exemplos nos apêndices.

FI - Subprograma para o cálculo do funcional de custo. Nesta sub-rotina, o valor do funcional de custo é calculado, em função de $x(t_f)$. A declaração para FI é:

```
SUBROUTINE FI(AC)
```

O usuário deve estabelecer a relação entre o valor do custo AC e $x(t_f)$, dado pela variável Xl(I), definida no COMMON.

Com este subprograma, a tarefa do usuário está terminada e as funções restantes estão implementadas no GRAMIN. No capítulo seguinte serão apresentados alguns exemplos de uso do GRAMIN, para o cálculo de entradas de controle nos espaços discutidos na seção 2.4.

CAPITULO 4

APLICAÇÕES

Com o objetivo de exemplificar o uso do algorítmo implementado neste trabalho, será abordado o problema do controle ótimo para a tensão de campo de um motor DC. As entradas de controle consideradas para este sistema pertencem a três tipos diferentes de espaço: funções contínuas, entradas amostradas e modulação por largura de pulso (MLP). No espaço de MLP, nos casos em que ocorrer saturação na variável de controle, será testado o algorítmo de Pagurek (49).

Como outro exemplo da aplicação do GRAMIN, serão determinados os valores ótimos para um compensador polo zero, considerando-se as entradas de controle no espaço de parâmetros. Ainda neste espaço, a aplicação do GRAMIN será extendida a um processo químico, através da determinação das constantes de reação na desidrogenação pirolítica do benzeno em difenil e trifenil.

4.1- CONTROLE ÓTIMO DE UM MOTOR DC.

O sistema considerado é um motor DC de aproximadamente 5hp, controlado pelo campo. Os parâmetros adotados para o motor são os mesmos usados por Hasdorff(28). O problema a ser resolvido é encontrar uma entrada de controle u(t), que produza uma mudança em degrau de 10 radianos no eixo do motor e minimize um certo critério de desempenho.

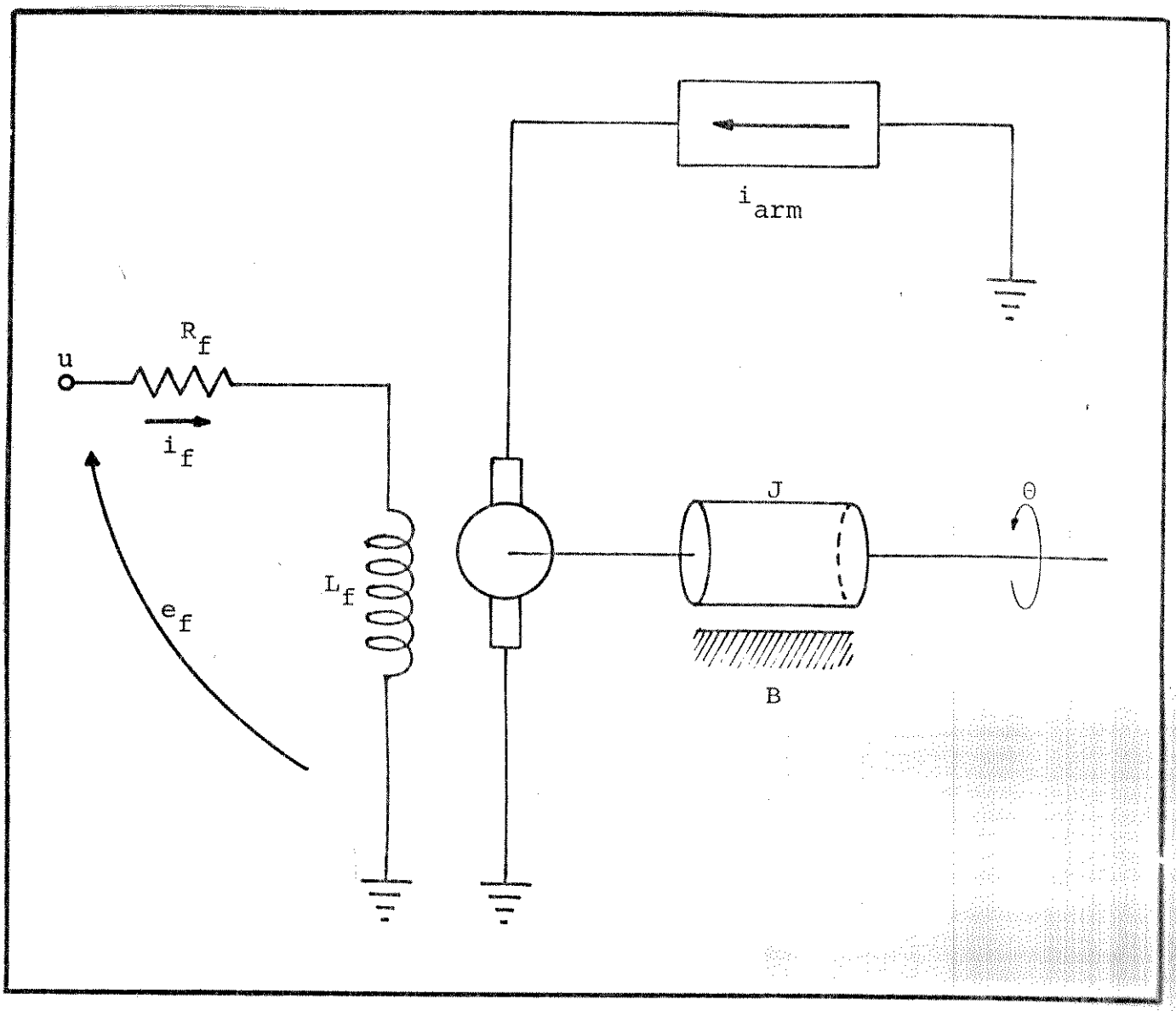


Fig.4.1 - Esquema e características do motor DC controlado pelo campo.

R_f = 10 Ω ; L_f = 10 h.

J = 2Kg.m^2; B = 0,66Kg.m^2/seg.

K_t = 10 N.m/A

4.1.1 - MODELAMENTO DO SISTEMA.

Na figura 4.1 está mostrado o esquema e características do motor DC controlado pelo campo.

Nesta figura temos:

R_f - resistência de campo;

L_f - indutância de campo;

i_f - corrente de campo;

e_f - tensão de campo;

i_{arm} - corrente de armadura;

B - coeficiente de amortecimento dinâmico;

J - momento de inércia;

θ - posição do eixo (em radianos);

K_t - constante de torque.

Para o sistema considerado as variáveis de estado são definidas por:

$$x_1 = \theta \tag{4.1.1}$$

$$x_2 = \dot{\theta} \tag{4.1.2}$$

$$x_3 = i_f \tag{4.1.3}$$

Com esta escolha de variáveis, o modelo do motor pode ser representado pelo seguinte conjunto de equações de estado:

$$\dot{x}_1 = x_2 \tag{4.1.4}$$

$$\dot{x}_2 = \left(\frac{B}{J}\right) x_2 + \left(\frac{K_t}{J}\right) x_3 \tag{4.1.5}$$

$$\dot{x}_3 = -\left(\frac{R_f}{L_f}\right) x_3 + \left(\frac{u}{L_f}\right) \tag{4.1.6}$$

Nestas expressões u é a entrada de controle e representa a

tensão (e_f) de campo.

O programa GRAMIN faz a determinação de um controle ótimo u*, que minimiza um dado critério de custo. Como nesta aplicação, estamos interessados que a posição do eixo do motor se aproxime de uma variação em degrau de 10rad, devemos formular um critério de custo que tenha um mínimo, quando a saída do sistema (x_1) apresentar tal variação. Desta forma o funcional de custo é definido como a integral do quadrado do êrro entre a posição real e a posição desejada. Nesta integral é adicionado um têrmo de penalização do custo $Ru^2(t)$, para limitar a energia aplicada na entrada do motor, evitando valores de tensão de entrada que não possam ser reproduzidos na prática. No funcional de custo são ainda acrescentados mais três termos de penalização com ponderações W_1, W_2, W_3, para garantir que o estado final seja atingido. Isso fornece um critério da forma:

$$J(u) = \int_{t_0}^{t_f} \left[(x_1 - 10)^2 + Ru^2\right] dt + \left[W_1(x_1 - 10)^2 + W_2x_2^2 + W_3x_3^2\right] \Big|_{t=t_f} \qquad (4.1.7)$$

Esta equação integral pode ser resolvida definindo uma nova variável de estado x_4, tal que:

$$\dot{x}_4 = (x_1 - 10)^2 + Ru^2 \quad \text{com} \quad x_4(0) = 0 \qquad (4.1.8)$$

Substituindo (4.1.8) em (4.1.7), o critério de custo pode ser escrito como:

$$J(u) = \phi(x(t_f)) = \left[x_4 + W_1(x_1 - 10)^2 + W_2x_2^2 + W_3x_3^2\right]_{t=t_f} \qquad (4.1.9)$$

O estado do sistema é agora definido pelo conjunto de equações (4.1.4), (4.1.5), (4.1.6) e (4.1.8), com estado inicial ($t=t_0$) dado por:

$$\underline{x}(0) = C = 0 \qquad (4.1.10)$$

Usando os parâmetros adotados por Hasdorff (vide figura 4.1), as equações de estado tornam-se:

$$\dot{x}_1 = x_2 \qquad (4.1.11)$$

$$\dot{x}_2 = -\frac{x_2}{3} + 5x_3 \qquad (4.1.12)$$

$$\dot{x}_3 = -x_3 + 0{,}1u \qquad (4.1.13)$$

$$\dot{x}_4 = (x_1 - 10)^2 + Ru^2 \qquad (4.1.14)$$

4.2- O CONTROLE ÓTIMO NO ESPAÇO DE FUNÇÕES CONTÍNUAS

No espaço de funções contínuas, a entrada de controle u(t) é uma função contínua no intervalo $[t_0, t_f]$. Para a aplicação do GRAMIN a este caso, o primeiro passo a ser executado é a obtenção da expressão para o gradiente do funcional de custo, que neste espaço é dado pela equação (2.4.31). Esta equação requer o cálculo de $\underline{f}_u$ e $\underline{\lambda}(t)$, onde $\underline{f}_u$ é definido por (2.4.15) e $\underline{\lambda}(t)$ é obtido através da solução da equação adjunta (2.4.32). Neste exemplo, considerando as equações de estado do motor temos:

$$\underline{f}_u^{\dagger} = [0 \quad 0 \quad 0{,}1 \quad 2Ru] \qquad (4.2.1)$$

$$\dot{\lambda}_1 = -2(x_1 - 10)\lambda_4 \qquad (4.2.2)$$

$$\dot{\lambda}_2 = -\lambda_1 + \frac{\lambda_2}{3} \qquad (4.2.3)$$

$$\dot{\lambda}_3 = -5\lambda_2 + \lambda_3 \qquad (4.2.4)$$

$$\dot{\lambda}_4 = 0 \qquad (4.2.5)$$

A condição terminal $\underline{\lambda}(t_f)$, para as equações (4.2.2) a (4.2.5),

por (2.4.32) fica:

$$\underline{\lambda}(t_f) = \begin{bmatrix} 2W_1(x_1 - 10) & 2W_2x_2 & 2W_3x_3 & 1 \end{bmatrix}^{\dagger} \qquad (4.2.6)$$

Portanto para este sistema e da equação (2.4.31), a expressão final para o gradiente é:

$$g(u(t)) = 0{,}1\lambda_3(t) + 2Ru(t)\lambda_4(t) \quad ; \; t \in [t_0, t_f] \qquad (4.2.7)$$

Através destas equações e seguindo os procedimentos da seção 3.2, o programa principal e os subprogramas auxiliares foram codificados, e estão listados no apêndice A3.

A seguir faremos uma análise comparativa dos resultados obtidos através do GRAMIN, com os resultados publicados por Hasdorff(28) para o motor DC, com entradas de controle no espaço de funções contínuas. Os resultados de Hasdorff servirão também como referência, para uma análise autoconsistente do comportamento do programa com alguns parâmetros tais como: número de iterações, número de reinicializações, intervalo de discretização, diferentes valores das funções de ponderação e diferentes estimativas iniciais para o controle u. Também serão testados os três tipos de Busca Unidimensional citadas no capítulo 3.

4.2.1 - ANÁLISE COMPARATIVA.

Para a execução do programa, consideramos em (4.1.14) R = 5 e para as constantes de ponderação (equação 4.1.7): $W_1 = W_2 = W_3 = 10\,000$. O controle é calculado no intervalo de tempo $[t_0, t_f] = [0 \text{ , } 5\text{seg}]$. A estimativa inicial para a entrada de controle é $u_0(t) = 0$, para $t \in [t_0, t_f]$ .Estes valores são os mesmos usados por Hasdorff na referência (28).

Na tabela 4.1, mostramos a convergência do quadrado da norma do gradiente $||g||^2$ e do valor do funcional de custo, para 12 iterações do GRAMIN. Estão incluidos também os resultados de Hasdorff para 11 iterações de seu programa. Nesta tabela, observa-se que o funcional de custo dado pelo GRAMIN, a exemplo dos valores de Hasdorff, converge monotonicamente à medida que se aumenta o número de iterações. O mesmo não acontece com a norma do gradiente, cujo valor atingido na 12º iteração, indica ser ainda possível uma redução no custo, com um número maior de iterações. Como o valor para o custo na 8º iteração do GRAMIN, é menor do que o obtido por Hasdorff na 11º iteração, para efeitos práticos o GRAMIN poderia ser interrompido após oito iterações. Observa-se ainda que, na 9º iteração, a razão entre os dois valores do custo é 1,02, apesar desta razão ter sido igual a 14,35 na primeira iteração.

A entrada de controle e a resposta do sistema encontradas pelo GRAMIN estão mostradas respectivamente, nas figuras 4.2b e 4.3b, para 10 iterações. Pode-se verificar a identidade destes resultados comparando-os com as figuras obtidas por Hasdorff (figuras 4.2a e 4.3a), também após 10 iterações.

4.2.2 - ANÁLISE AUTOCONSISTENTE.

Tendo em vista, a concordância dos resultados mostrada na análise comparativa, vamos usar este mesmo exemplo, para estudar a sensitividade do programa à variação de alguns parâmetros, que consideramos importantes para a aplicação do método do Gradiente Conjugado à solução dos problemas de controle ótimo. A análise será feita de uma maneira autoconsistente, significando que apenas um parâmetro sofrerá variação, enquanto os demais permanecerão inalterados. Em cada caso estudado, será empregada a Busca Unidimensional na forma mais

TABELA 4.1: CONVERGÊNCIA DO GRADIENTE CONJUGADO PARA O CASO DE FUNÇÕES CONTÍNUAS.

NPTOS=150 W_i=10 000 IRREST=1 $u_0(t)$=0

GRAMIN			HASDORFF	
ITERAÇÃO	$\|\|g\|\|^2$	custo	$\|\|g\|\|^2$	custo
1	2,44E+8	6,97E+4	8,54E+10	1,00E+6
2	2,00E+8	4,00E+3	2,45E+8	7,00E+4
3	1,05E+5	1,86E+3	6,06E+6	9,09E+3
4	8,11E+6	1,76E+3	3,34E+5	1,85E+3
5	3,14E+6	1,11E+3	2,14E+5	1,85E+3
6	4,00E+5	1,o8E+3	2,22E+6	1,65E+3
7	2,69E+6	1,04E+3	5,96E+6	1,63E+3
8	1,62E+5	9,69E+2	1,34E+7	1,18E+3
9	1,54E+3	9,68E+2	2,69E+4	9,84E+2
10	1,05E+3	9,68E+2	5,08E+3	9,84E+2
11	2,00E+4	9,68E+2	1,70E+1	9,84E+2
12	1,95E+3	9,67E+2	*	*

* Dados não disponíveis.

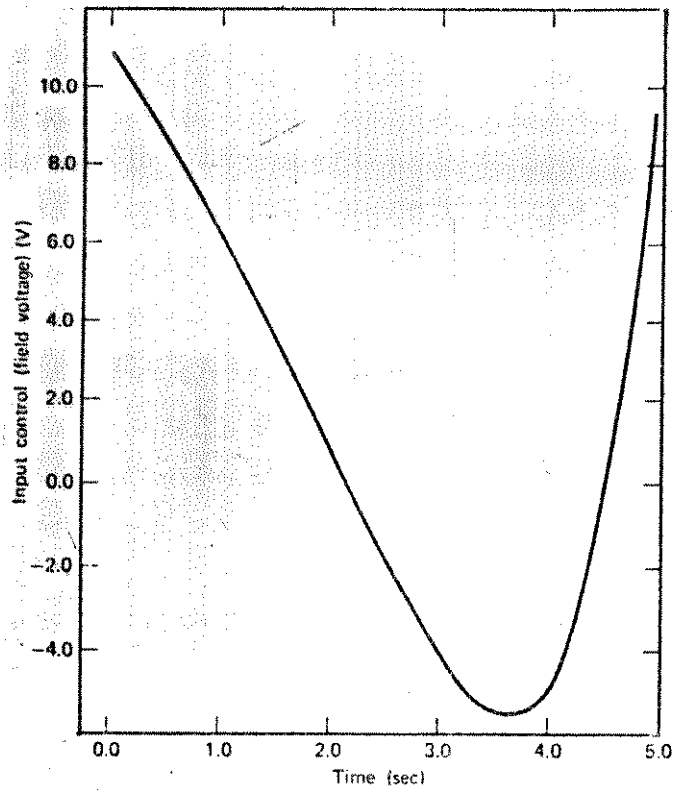


Fig. 4.2a - Entrada para o motor DC obtida por Hasdorff (28) caso de funções contínuas.

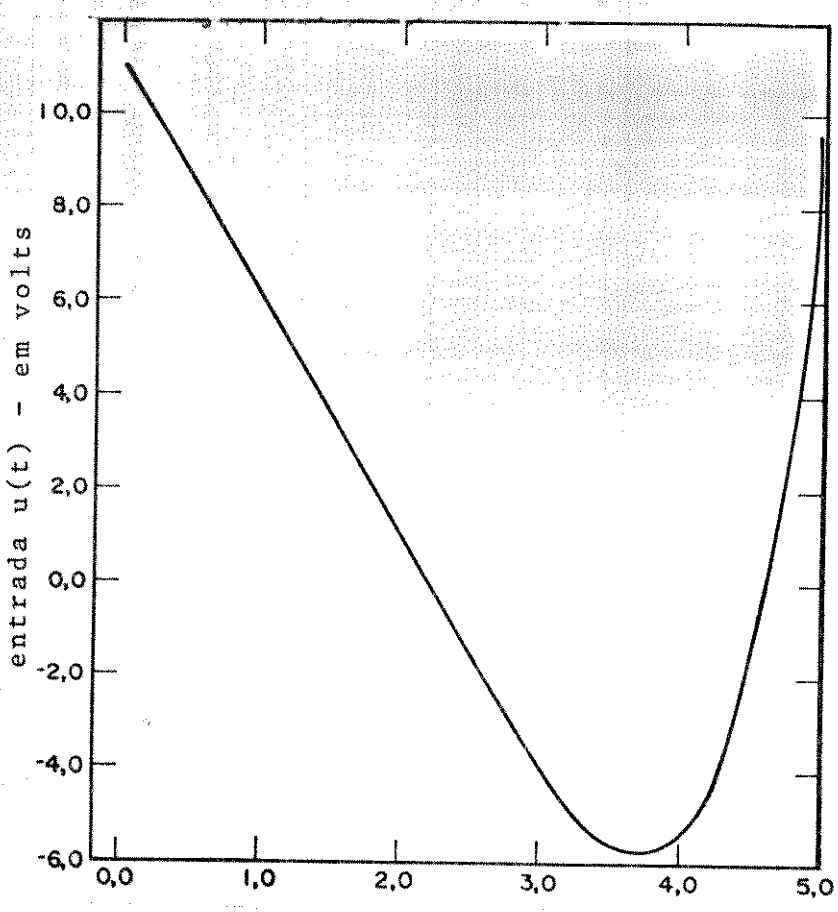


Fig. 4.2b - Entrada para o motor DC obtida pelo GRAMIN.
caso de funções contínuas.

NPTOS = 150

NITER = 10

IRREST = 1

$u_0(t) = 0$

$W_i = 100\,000$

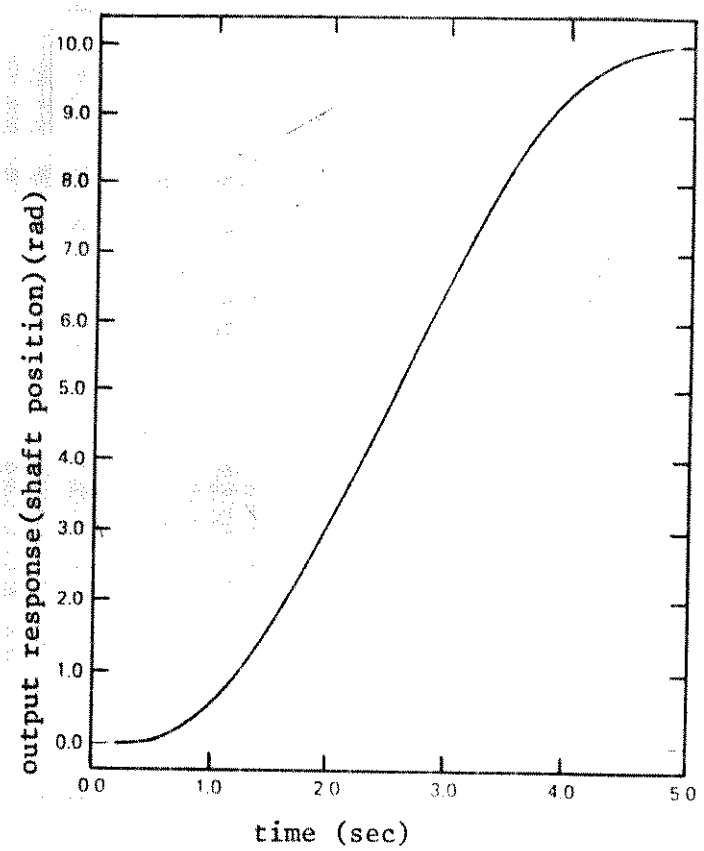


Fig. 4.3a - Posição do eixo do motor DC, obtido por Hasdorff (28), com 10 iterações.
caso de funções contínuas.

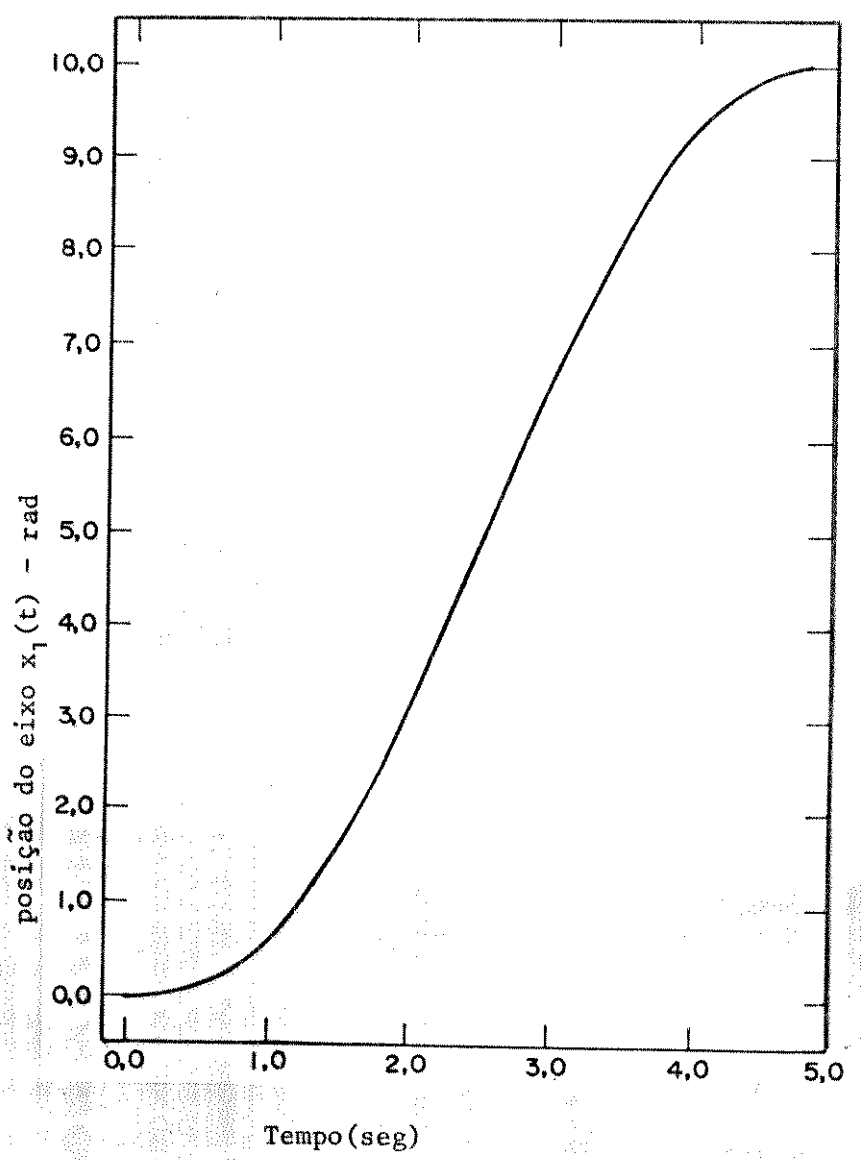


Fig. 4.3b - Posição do eixo do motor, $x_1(t)$ fornecida pelo GRAMIN.
caso de funções contínuas.
NPTOS = 150
NITER = 10
IRREST = 1
$u_0(t) = 0$
$W_i = 100\,000$

completa, exceto no caso da análise da influência dos diferentes tipos de buscas.

1 - Estudo da variação do número de iterações.

As tabelas 4.2.a, 4.2.b e 4.2.c, mostram a cada iteração e a cada segundo, as variáveis de estado e as entradas de controle do motor. Para efeito de discretização, o intervalo de tempo $[t_0, t_f]$ foi dividido em 150 sub-intervalos (NPTOS=150). Para as funções de ponderação, adotou-se $W_1 = W_2 = W_3 = 10\,000$. Neste caso, não se fez reinicialização (ou restart) no GRAMIN (IRREST=1). E como estimativa inicial para a entrada de controle, considerou-se $u_0(t)=0$.

Podemos estudar mais facilmente estas tabelas, considerando que conhecemos: o estado final desejado ($x_1(t_f)=10$rad, $x_2(t_f)=0$ rad/seg e $x_3(t_f)=0$A), os valores do funcional de custo à cada iteração; bem como as curvas da entrada de controle e da resposta do sistema , obtidos por Hasdorff (respectivamente tabela 4.1, figuras 4.2a e 4.3a).

Através da análise do estado final do sistema, observa-se que $x_1(t_f)$, à partir da terceira iteração, difere em menos de 1% do seu valor na 12º iteração. Com relação a $x_2(t_f)$, as diferenças mais significativas ocorrem nas primeiras 4 iterações, atingindo o valor de 0,02 na oitava e mantendo-se em torno deste valor até a 12º iteração. As principais alterações em $x_3(t_f)$ ocorrem até a 5º iteração, e à partir da oitava estabiliza-se em -0,03.

Ainda das tabelas 4.2, pode-se notar que a resposta do sistema $x_1(t)$ apresenta a mesma forma mostrada na figura 4.3a, já à partir da primeira iteração, enquanto que a entrada de controle $u(t)$, passa a ter o comportamento indicado na figura 4.2a, apenas após a 5º iteração.

Destas observações, vemos que as alterações mais relevantes

TABELA 4.2.a - ESTUDO DA VARIAÇÃO DO NÚMERO DE ITERAÇÕES: ESPAÇO DE FUNÇÕES CONTÍNUAS

NPTOS = 150 W_i = 10 000 IRREST = 1 $u_0(t) = 0$

NITER = 1						NITER = 2				
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,26923	0,71561	0,26778	190,693	3,94655	0,52172	1,35571	0,48265	412,023	6,25797
2	1,54338	1,81139	0,31784	338,243	3,06889	2,78748	3,03720	0,43449	587,549	2,26632
3	3,77459	2,57043	0,27263	426,591	1,96780	6,15227	3,42691	0,14476	627,754	-2,15126
4	6,50792	2,80671	0,18148	461,193	0,76716	9,11397	2,26372	-0,20846	713,348	-5,19068
5	9,30671	2,53627	0,07897	467,026	0,00116	10,40642	0,26271	-0,30187	804,337	-0,52665
NITER = 3						NITER = 4				
1	0,51768	1,34382	0,47730	405,909	6,14745	0,52244	1,35481	0,48010	410,331	6,14510
2	2,75681	2,99257	0,42303	575,685	2,08480	2,77365	3,00090	0,41997	577,933	1,97519
3	6,05278	3,33112	0,12657	616,744	-2,39940	6,06385	3,30688	0,11611	619,229	-2,55616
4	8,88764	2,10506	-0,23164	713,495	-5,44405	8,85283	2,04064	-0,24069	719,664	-5,40931
5	9,98696	0,05520	-0,31919	812,439	-0,53313	9,89419	0,03852	-0,28819	805,899	0,51570

TABELA 4.2.b - ESTUDO DA VARIAÇÃO DO NÚMERO DE ITERAÇÕES: ESPAÇO DE FUNÇÕES CONTÍNUAS

NPTOS = 150 W_i = 10 000 IRREST = 1 $u_0(t) = 0$

NITER = 5						NITER = 6				
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,57555	1,48407	0,51921	468,680	6,41268	0,58015	1,49562	0,52302	474,332	6,45170
2	3,00402	3,19032	0,42121	631,434	1,42112	3,02624	3,21190	0,42321	637,656	1,40860
3	6,41419	3,32074	0,06269	676,490	-3,52054	6,45663	3,33700	0,06129	682,851	-3,56481
4	9,07612	1,78829	-0,29112	799,869	-5,09690	9,12762	1,79026	-0,29323	807,397	-5,08435
5	9,91197	0,10858	-0,06146	879,570	7,97296	9,96654	0,12341	-0,04980	890,867	8,35758
NITER = 7						NITER = 8				
1	0,58864	1,51696	0,53002	484,863	6,52174	0,69112	1,54777	0,53971	500,083	6,60215
2	3,06701	3,25092	0,42648	649,057	1,37395	3,12363	3,30072	0,42846	664,172	1,26808
3	6,53235	3,36231	0,05704	694,900	-3,68254	6,62555	3,37615	0,04537	712,189	-3,93338
4	9,31049	1,77754	-0,30188	824,753	-5,15819	9,28074	1,71548	-0,32108	853,966	-5,31975
5	10,02397	0,10138	-0,04174	914,343	8,80582	10,00537	0,02079	-0,02948	956,017	9,60993

TABELA 4.2.c - ESTUDO DA VARIAÇÃO DO NÚMERO DE ITERAÇÕES: ESPAÇO DE FUNÇÕES CONTÍNUAS

NPTOS = 150 W_i = 10 000 IRREST = 1 $u_0(t) = 0$

NITER = 9						NITER = 10				
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,60107	1,54759	0,53961	499,963	6,59923	0,60102	1,54745	0,53955	499,886	6,59844
2	3,12303	3,29966	0,42813	663,892	1,26223	3,12272	3,29930	0,42807	663,780	1,26178
3	6,62311	3,37344	0,04480	712,023	-3,94102	6,62238	3,37298	0,04477	711,927	-3,94106
4	9,27461	1,71087	-0,32170	854,168	-5,32427	9,27344	1,71046	-0,32171	854,071	-5,32417
5	9,99507	0,01602	-0,02936	956,502	9,62901	9,99351	0,01569	-0,02939	956,389	9,62490
NITER = 11						NITER = 12				
1	0,60041	1,54585	0,53897	499,045	6,59107	0,59989	1,54447	0,53848	498,327	6,58530
2	3,11947	3,29588	0,42769	662,796	1,26322	3,11676	3,29320	0,42749	662,009	1,26805
3	6,61585	3,37038	0,04509	710,848	-3,92949	6,61102	3,36949	0,04578	709,897	-3,91257
4	9,26599	1,71155	-0,32080	852,359	-5,31445	9,26264	1,71568	-0,31944	850,507	-5,30128
5	9,98871	0,01831	-0,03004	953,949	9,58095	9,99145	0,02407	-0,03086	951,133	9,52274

nas variáveis de estado e na entrada de controle, ocorrem nas primeiras 5 iterações do método. Este fato, também pode ser notado na tabela 4.1 com relação ao funcional de custo, sendo que à partir da 8º iteração o método poderia ser interrompido. Isto é vantajoso, pois o tempo de computação é diretamente proporcional ao número de iterações.

Fletcher e Reeves em (44), sujerem que a velocidade de convergência do gradiente conjugado, pode ser aumentada se a cada n+1 iterações for feita uma reinicialização do método, onde n representa o número de variáveis de estado do sistema. Como nas tabelas 4.1 e 4.2 observa-se que as principais alterações ocorrem até a 5º iteração, torna-se interessante investigar a influência das reinicializações sobre a convergência do algorítmo. Isto será feito na próxima análise.

2 - Influência do número de reinicializações.

A tabela 4.3 foi contruida, tomando-se os resultados com diferentes reinicializações. Estas foram feitas após 5 iterações, e os dados mostram a evolução do sistema a cada segundo.

A variável IRREST, é uma variável de contagem para o número de chamadas da sub-rotina GRAMI. Assim sendo, quando IRREST=1, significa que esta sub-rotina foi chamada pela primeira vez. Portanto o número de reinicializações do método equivale a IRREST-1.

Pode-se observar na tabela 4.3, que o valor do funcional de custo é $9{,}72 \times 10^2$, com uma reinicialização (IRREST=2). Em termos de tempo de computação, 5 iterações com uma reinicialização equivale, aproximadamente, a 10 iterações sem restart. Com este número de iterações o valor do funcional de custo é $9{,}68 \times 10^2$ (tabela 4.1), sendo portanto mais vantajoso, nestas condições, não se reinicializar,

TABELA 4.3 - INFLUÊNCIA DO NÚMERO DE REINICIALIZAÇÕES SOBRE A CONVERGÊNCIA DO MÉTODO:

FUNÇÕES CONTÍNUAS

NPTOS = 150 W_i = 10 000 NITER = 5 $u_0(t) = 0$

IRREST = 1			J(u) = 1112,731			IRREST = 2		J(u) = 971,717		
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,57555	1,48407	0,51921	468,680	6,41268	0,59006	1,52018	0,53083	486,323	6,52175
2	3,00402	3,19032	0,42121	631,434	1,42112	3,07216	3,25405	0,42601	650,124	1,35357
3	6,41419	3,32074	0,06269	676,490	-3,52054	6,53784	3,35871	0,05471	696,398	-3,73128
4	9,07612	1,78829	-0.29112	799,869	-5,09690	9,20522	1,75710	-0,30846	830,541	-5,29682
5	9,91197	0,10858	-0,06146	879,570	7,97296	9,97655	0,03081	-0,06278	917,309	8,36781
IRREST = 4			J(u) = 964,136			IRREST = 6		J(u) = 964,067		
1	0,59997	1,54364	0,53747	497,482	6,55454	0,60065	1,54497	0,53764	497,998	6,54815
2	3,11276	3,28575	0,42593	660,177	1,26844	3,11434	3,28571	0,42543	660,324	1,26102
3	6,59915	3,36390	0,04754	707,449	-3,85075	6,59960	3,36207	0,04735	707,538	-3,84446
4	9,25283	1,72653	-0,31545	845,218	-5,28969	9,25196	1,72630	-0,31487	844,790	-5,28127
5	9,99298	0,02042	-0,04550	938,770	8,98228	9,99279	0,02035	-0,04887	937,437	8,91795

mas fazer 10 iterações. Já com 5 iterações e 3 reinicializações , o custo cai para $9,64 \times 10^2$, e os quatro últimos dados da tabela 4.1 parecem indicar que após 20 iterações o custo convergiria para um valor maior que este, uma vez que a convergência apresentada é lenta. Neste caso é mais vantajoso fazer reinicializações do método. Nota-se ainda, que para um número superior a três reinicializações, a redução no custo não é significativa, considerando-se o aumento adicional no tempo de computação.

3 - Diferentes intervalos de discretização.

Como visto no capítulo 3, a variável NPTOS, define o número de sub-intervalos existentes entre os instantes t_0 e t_f. Durante a execução do programa estes sub-intervalos determinam o passo no Runge-Kutta, e portanto o tempo de computação é diretamente proporcional a variável NPTOS. Além disso, para a execução do programa, é necessário armazenar (NPTOS+1)X(2n+5) componentes vetoriais, onde n é a dimensão do vetor de estado do sistema. Assim sendo, tanto para aumentar a velocidade de computação, como para minimizar a quantidade de memória requerida, é interessante que a discretização do intervalo de tempo $[t_0, t_f]$ seja adequadamente escolhida. Por exemplo, um programa com NPTOS=200, requer 10 vezes mais memória para armazenar os vetores do sistema, além de ser 10 vezes mais lento, do que outro com NPTOS=20.

Na tabela 4.4 estão os resultados obtidos , tomando como número de sub-intervalos os seguintes valores: 20, 50, 150 e 200. Pode-se observar que os valores do custo, crescem com NPTOS. Isto ocorre, porque neste tipo particular de função, à medida que se diminui o intervalo de integração a variável x_4, que contribui diretamente para o custo, aumenta de valor. Desta forma, só tem significado com-

TABELA 4.4 - INFLUÊNCIA DO INTERVALO DE DISCRETIZAÇÃO: FUNÇÕES CONTÍNUAS

NITER = 10 W_i = 10 000 IRREST = 1 $u_0(t) = 0$

NPTOS = 20				J(u) = 898,775		NPTOS = 50			J(u) = 934,897	
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,55540	1,43727	0,50644	448,070	6,83561	0,57329	1,47585	0,51324	462,810	6,49419
2	2,91451	3,11056	0,42063	609,781	2,31447	2,98554	3,18243	0,43002	629,668	1,46754
3	6,27005	3,31002	0,08867	651,402	-2,43163	6,41532	3,36942	0,07716	673,473	-3,55558
4	8,99128	1,91772	-0,25900	758,058	-5,11847	9,14386	1,85882	-0,29279	801,024	-5,40235
5	9,97936	0,04927	-0,00389	870,089	6,82194	9,99432	0,01151	-0,03588	920,374	8,61115
NPTOS = 150				J(u) = 967,905		NPTOS = 200			J(u) = 968,379	
1	0,60102	1,54745	0,53955	499,886	6,59844	0,60669	1,56482	0,54571	580,848	6,64157
2	3,12272	3,29930	0,42807	663,790	1,26178	3,15965	3,31543	0,42437	669,445	1,31886
3	6,62238	3,37298	0,04477	711,927	-3,94106	6,65110	3,35975	0,03831	718,227	-3,86376
4	9,27344	1,71046	-0,32171	854,071	-5,32417	9,28012	1,68728	-0,31944	857,264	-5,21192
5	9,99351	0,01569	-0,02939	956,389	9,62490	9,99338	0,01990	-0,04593	942,880	9,55448

parar os valores de custo, quando se trabalha com o mesmo número de intervalos de discretização.

Analisando o estado final do sistema, e a forma da entrada de controle u(t), nota-se que mesmo com NPTOS=20, chega-se a uma boa aproximação dos resultados de Hasdorff. Este comportamento é desejável, principalmente quando se tem em vista a implementação em microcomputadores e controle em tempo real.

4 - Influência da função de ponderação.

Os resultados apresentados por Hasdorff, foram obtidos adotando-se o valor de 10 000, para as funções de ponderação dos termos de penalização no funcional de custo (equação 4.1.7). Entretanto, o autor não faz referência à respeito desta escolha. Por este motivo, na tabela 4.5, investigamos o comportamento do sistema para os seguintes valores de W_i (i=1,2,3): 0, 100, 10 000 , 100 000 .

Com W_i=0, não há penalização no funcional de custo, e neste caso a variável x_4 é o próprio funcional. Pode-se observar que a saída do sistema ficou em 5,1 radianos, sendo este a metade do valor desejado. O controle u(t), também apresenta uma característica diferente daquela mostrada na figura 4.2a.

Para W_i=100, tanto a resposta do sistema, o controle e o estado final já se aproximam mais dos resultados desejados, porém estes valores não são satisfatórios.

Os melhores resultados foram obtidos com W_i=10 000 e 100 000 , sendo que para W_i=100 000 o estado final das variáveis x_1, x_2 e x_3 , mais se aproxima do estado final especificado. Entretanto no último caso, a variável x_4 apresenta um valor maior que o anterior (com W_i= 10 000). Isto ocorre, porque o controle gerado com W_i=100 000 , requer mais energia, acarretando o aumento do funcional de custo. Portanto

TABELA 4.5 - DIFERENTES VALORES DA FUNÇÃO DE PONDERAÇÃO: FUNÇÕES CONTÍNUAS.

NPTOS = 150 IRREST = 1 NITER = 10 $u_0(t) = 0$

$W_i = 0$			$J(u) = 401,445$			$W_i = 100$		$J(u) = 691,263$		
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,20877	0,53582	0,18644	147,151	2,32329	0,45897	1,18484	0,41654	335,488	5,26901
2	1,09092	1,17786	0,17131	251,041	1,15322	2,41939	2,61389	0,37101	484,338	2,03872
3	2,42998	1,43226	0,10742	322,569	0,41736	5,32482	2,99198	0,15237	528,343	-1,06694
4	3,84233	1,35048	0,05166	370,046	0,06996	8,01493	2,22402	-0,09365	565,672	-3,03408
5	5,11432	1,09825	0,01925	401,445	0,00000	9,60937	0,88007	-0,15532	596,140	0,11059
$W_i = 10\,000$			$J(u) = 967,905$			$W_i = 100\,000$		$J(u) = 1027,468$		
1	0,60102	1,54745	0,53955	499,886	6,59844	0,61103	1,57328	0,54845	513,328	6,69545
2	3,12272	3,29930	0,42807	663,790	1,26178	3,17271	3,34732	0,43128	678,191	1,17523
3	6,62238	3,37298	0,04477	711,927	-3,94106	6,71015	3,39018	0,03376	728,983	-4,19586
4	9,27344	1,71046	-0,32171	854,071	-5,32417	9,34256	1,65485	-0,33832	882,050	-5,37597
5	9,99351	0,01569	-0.02939	956,389	9,62490	10,00003	-0,00265	-0,01284	1010,278	11,24459

dentre os valores estudados o controle com W_i=10 000 , apresenta o melhor desempenho, em vista de requerer menos energia e chegar a um estado final satisfatório.

5 - Diferentes valores iniciais para a entrada de controle.

A tabela 4.6, mostra a evolução no tempo das variáveis de estado e da entrada de controle do sistema, para diferentes valores iniciais do controle $u_0(t)$. Estes dados indicam, que após 10 iterações, o estado final, a resposta do sistema, a entrada de controle e o funcional de custo, praticamente não dependem do valor escolhido para $u_0(t)$. Este tipo de comportamento é útil, quando não se conhece, a priori, qual a forma da entrada de controle.

6 - Análise das diferentes formas de buscas.

A implementação do algorítmo da BUSCA, foi descrita na seção 3.1.4, quando então comentou-se a existência de três tipos de busca:

I - Busca incompleta (a) , a mais simples de todas;

II - Busca incompleta (b) , refina o intervalo de interpolação.

III - Busca completa, além de refinar o intervalo de interpolação, faz uma divisão do passo inicial.

A eficiência de cada uma destas buscas foi testada, sob diferentes condições, nas tabelas 4.7, 4.8, e 4.9 . Na tabela 4.7 observa-se , que não há diferença significativa entre os resultados obtidos com as 3 buscas. Porém o tempo de computação é menor para a busca incompleta (a), e a execução do programa com esta busca, requer metade do tempo gasto com a busca mais completa.

Na tabela 4.8, o número de intervalos de dicretização foi diminuído (NPTOS=50), com o objetivo de verificar o desempenho das

TABELA 4.6 - INFLUÊNCIA DOS DIFERENTES VALORES INICIAIS DA ENTRADA DE CONTROLE: FUNÇÕES CONTÍNUAS.

NPTOS = 150 W_i = 10 000 IRREST = 1 NITER = 10

$u_0(t) = -5$				J(u) = 964,417		$u_0(t) = 0$			J(u) = 967,905	
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,59717	1,53772	0,53633	494,957	6,56733	0,60102	1,54745	0,53955	499,886	6,59844
2	3,10440	3,28256	0,42727	658,754	1,29930	3,12272	3,29930	0,42807	663,790	1,26178
3	6,59187	3,36896	0,04910	705,990	-3,84870	6,62238	3,37298	0,04477	711,927	-3,94106
4	9,25198	1,73247	-0,31597	844,591	-5,31760	9,27344	1,71046	-0,32171	854,071	9,62490
5	9,99480	0,01975	-0,04620	938,898	9,00907	9,99351	0,01569	-0,02939	956,389	9,62490
$u_0(t) = 5$				J(u) = 966,108		$u_0(t) = 50$			J(u) = 964,863	
1	0,59960	1,54400	0,53851	498,196	6,59179	0,59388	1,53006	0,53428	491,377	6,56209
2	3,11665	3,29448	0,42816	662,269	1,27964	3,09168	3,27265	0,42782	655,818	1,33354
3	6,61373	3,37372	0,04650	710,079	-3,91090	6,57468	3,37004	0,05161	702,757	-3,81709
4	9,26985	1,71933	-0,32012	851,350	-5,33415	9,24118	1,74276	-0,31457	840,884	-5,33549
5	9,99724	0,01432	-0,03638	950,751	9,39553	9,99263	0,02156	-0,05070	933,965	8,88123

TABELA 4.7 - ANÁLISE DAS DIFERENTES FORMAS DE BUSCAS: FUNÇÕES CONTÍNUAS

NITER = 10 NPTOS = 150 W_i = 10 000 IRREST = 1 $u_0(t) = 0$

I - BUSCA INCOMPLETA (a) J(u) = 967,919

t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,60102	1,54747	0,53956	499,895	6,59848
2	3,12275	3,29932	0,42807	663,798	1,26169
3	6,62243	3,37297	0,04476	711,937	-3,94120
4	9,27346	1,71042	-0,32172	854,084	-5,32408
5	9,99351	0,01571	-0,02935	956,418	9,62617

II - BUSCA INCOMPLETA (b) J(u) = 967,905

x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
0,60102	1,54745	0,53955	499,896	6,59844
3,12272	3,29930	0,42807	663,790	1,26178
6,62238	3,37298	0,04477	711,927	-3,94106
9,27344	1,71046	-0,32171	854,071	-5,32417
9,99351	0,02358	-0,02939	956,389	9,62490

III - BUSCA COMPLETA (c) J(u) = 967,905

1	0,60102	1,54745	0,53955	499,886	6,59844
2	3,12272	3,29930	0,42807	663,790	1,26178
3	6,62238	3,37298	0,04477	711,927	-3,94106
4	9,27344	1,71046	-0,32171	854,071	-5,32417
5	9,99351	0,01568	-0,02939	956,389	9,62490

TEMPO DE CPU (PDP 10)

I - BUSCA INCOMPLETA (a): 31 seg

II - BUSCA INCOMPLETA (b): 57 seg

III - BUSCA COMPLETA (c): 65 seg

TABELA 4.8 - ANÁLISE DAS DIFERENTES BUSCAS COM NPTOS = 50.

FUNÇÕES CONTÍNUAS

I - BUSCA INCOMPLETA (a)				J(u) = 934,897		II - BUSCA INCOMPLETA (b)			J(u) = 934,905	
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,57328	1,47583	0,51323	462,800	6,49417	0,57332	1,47594	0,51326	462,846	6,49425
2	2,98551	3,18240	0,43002	629,661	1,46767	2,98567	3,18251	0,43000	629,694	1,46695
3	6,41528	3,36944	0,07717	673,465	-3,55542	6,41547	3,36934	0,07711	673,505	-3,55628
4	9,14386	1,85887	-0,29278	801,015	-5,40260	9,14382	1,85857	-0,29281	801,060	-5,40126
5	9,99433	0,01141	-0,03597	920,337	8,60883	9,99421	0,01193	-0,03551	920,534	8,62129
II - BUSCA COMPLETA (c)				J(u) = 934,897						
1	0,57329	1,47585	0,51324	462,810	6,49419					
2	2,98554	3,18243	0,43002	629,668	1,46754					
3	6,41532	3,36942	0,07716	673,473	1,46754					
4	9,14386	1,85882	-0,29279	801,024	-5,40235					
5	9,99432	0,01151	-0,03588	920,374	8,61115					

W_i = 10 000

NITER = 10

IRREST = 1

$u_0(t)$ = 0

TABELA 4.9 - ANÁLISE DAS DIFERENTES BUSCAS , COM NITER = 5.

FUNÇÕES CONTÍNUAS

I - BUSCA INCOMPLETA (a) J(u) = 1112,407						II - BUSCA INCOMPLETA (b) J(u) = 1112,731				
t (seg)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	x_4	u(t) (volt)
1	0,57561	1,48421	0,51925	468,745	6,41301	0,57555	1,48407	0,51921	468,680	6,41268
2	3,00426	3,19054	0,42122	631,496	,42063	3,00402	3,19032	0,42121	631,434	1,42112
3	6,41459	3,32079	0,06264	676,560	-3,52157	6,41419	3,32074	0,06269	676,490	-3,52054
4	9,07642	1,78804	-0,29119	799,976	-5,09713	9,07612	1,78829	-0,29112	799,869	-5,09690
5	9,91198	0,10844	-0,06132	879,734	7,97836	9,91197	0,10858	-0,06146	879,570	7,97296
III - BUSCA COMPLETA (c) J(u) =1112,731										
1	0,57555	1,48407	0,51921	468,680	6,41268					
2	3,00402	3,19032	0,42121	631,434	1,42112					
3	6,41419	3,32074	0,06269	676,490	-3,52054					
4	9,07612	1,78829	-0,29112	799,869	-5,09690					
5	9,91197	0,10858	-0,06146	879,570	7,97296					

NPTOS = 150

W_i = 10 000

IRREST = 1

$u_0(t)$ = 0

buscas em tais condições. Novamente nenhuma diferença relevante foi notada. O mesmo se deu em relação aos resultados da tabela 4.9, onde desta vez foi reduzido o número de iterações, mantendo-se os demais parâmetros.

Destas observações concluímos, que no espaço de Funções Contínuas e neste exemplo em particular, é mais interessante aplicar a busca incompleta (a), considerando que ela é mais rápida, e fornece os mesmos resultados que as demais.

4.3- OBTENÇÃO DO CONTROLE ÓTIMO NO ESPAÇO DE ENTRADAS AMOSTRADAS.

Neste espaço, vamos resolver novamente o problema de controle ótimo do motor DC (figura 4.1), para encontrar a entrada de controle, que produza uma mudança em degrau de 10 radianos. Entretanto aqui a entrada de controle tem a forma mostrada na figura 2.4. Todos os parâmetros que caracterizam o motor são mantidos, bem como as equações de estado (4.1.11 - 4.1.14), o funcional de custo (4.1.9), a equação adjunta (4.2.2 - 4.2.5) e sua condição terminal (4.2.6) .

No espaço de entradas amostradas, o gradiente é fornecido pela equação (2.4.42), onde a expressão para $\underline{f}_u^{\dagger}$ é a mesma de (4.2.1).

Com estes dados a k-ésima componente do gradiente fica:

$$g_k = \int_{t_0+KT}^{t_0+\overline{K+1}\,T} \left[0{,}1\lambda_3(t) + 2Ru_k\lambda_4(t) \right] dt \qquad (4.3.1)$$

com $k = 0,1,2,\ldots,7$.

Para este exemplo, o programa e as sub-rotinas desenvolvidas estão listados no apêndice A4.

RESULTADOS

Nas figuras 4.4b e 4.5b tem-se a entrada de controle e a res-

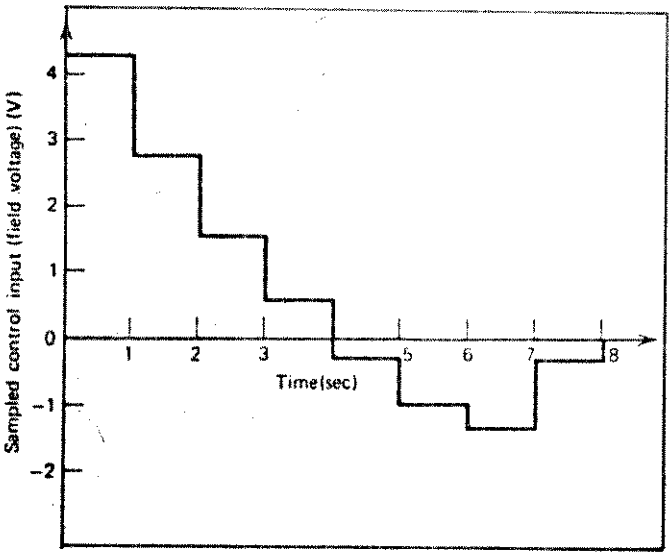


Fig. 4.4a - Entradas de controle amostradas , para o motor DC Hasdorff (28).
espaço de entradas amostradas

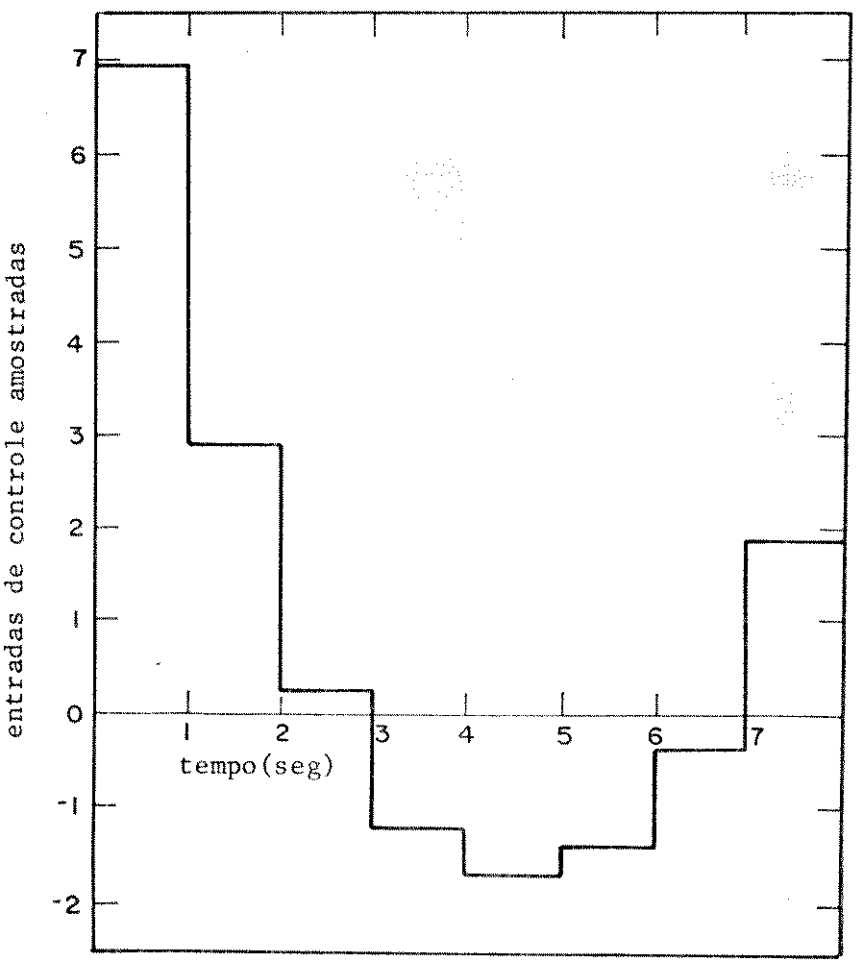


Fig. 4.4b - Entradas amostradas obtidas através do GRAMIN, para o motor DC.
NPTOS =160
NITER = 12
IRREST = 1
W_i = 1000
$u_0(k)$ = 0,1,...,7

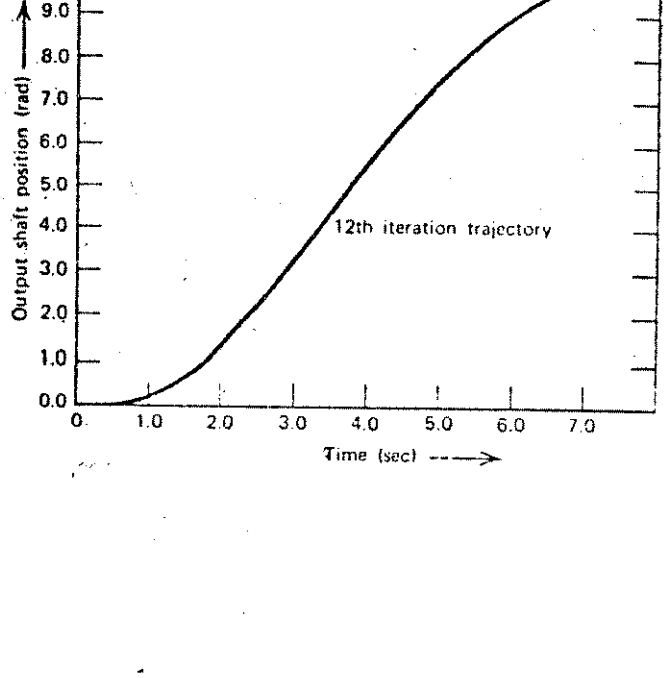


Fig.4.5a - posição do eixo do motor DC em função do tempo - Hasdorff (28).

caso: entradas amostradas.

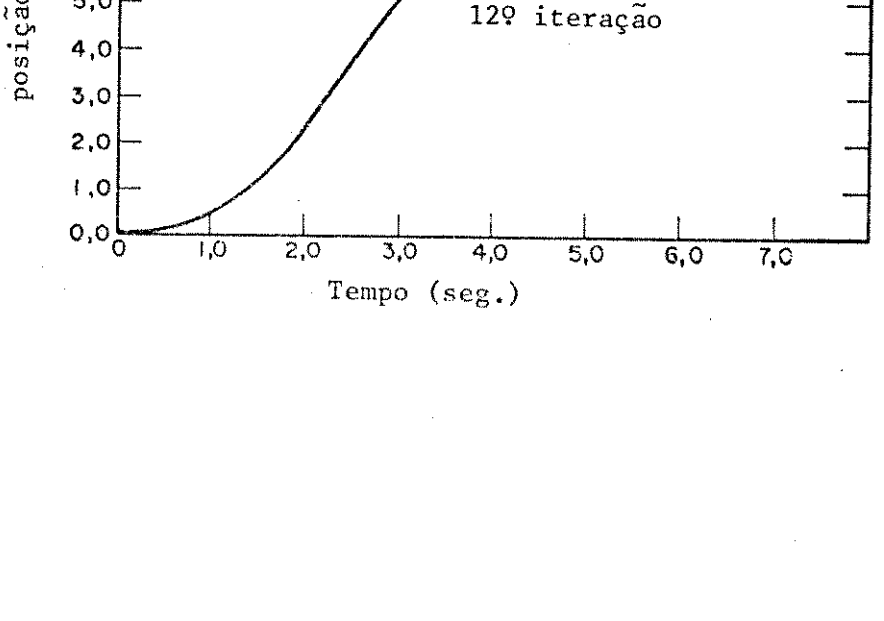


Fig.4.5b - Posição $x_1(t)$ do eixo do motor DC, em função do tempo - GRAMIN .

caso: entradas amostradas.

NPTOS = 160

NITER = 12

IRREST = 1

W_i = 1000

$u_0(k) = 0$; $k = 0,1,\ldots,7$

TABELA 4.10 - CONVERGÊNCIA DO GRADIENTE CONJUGADO PARA O CASO DE ENTRADAS AMOSTRADAS.

NPTOS = 160 W_i = 1000 IRREST = 1 NITER = 12

$u_0(k) = 0$; $k = 0,1,2,\ldots,7$

	GRAMIN		HASDORFF	
iteração	$\|\|g\|\|^2$	custo	$\|\|g\|\|^2$	custo
1	1,062E+6	2,149E+3	1,107E+6	2,268E+3
2	1,104E+5	3,539E+2	6,222E+6	7,417E+2
3	6,350E+2	3,499E+2	6,753E+2	4,739E+2
4	2,155E+4	3,490E+2	3,320E+4	4,726E+2
5	7,764E+3	3,381E+2	4,116E+3	4,509E+2
6	1,696E+4	3,371E+2	5,412E+2	4,507E+2
7	5,990E+4	2,812E+2	4,122E+3	4,501E+2
8	3,584E+2	2,790E+2	*	*
9	1,730E+3	2,789E+2	*	*
10	2,968E+3	2,775E+2	*	*
11	6,401E+2	2,774E+2	*	*
12	5,080E+3	2,752E+2	3,163E+1	4,494E+2

(*) Dados não disponíveis.

posta do sistema obtidos utilizando o GRAMIN. Os resultados foram tomados após 12 iterações, considerando que a entrada de controle é amostrada oito vezes no intervalo de tempo $[t_0=0, t_f=8seg]$. Para as funções de ponderação adotou-se: $W_1= W_2= W_3= 1000$; e no Runge-Kutta foram utilizados 160 intervalos para a integração. Não foram feitas reinicializações no método, e a estimativa inicial do controle foi $u_0(k)=0$, para $k=0,1,2,\ldots,7$. Os resultados obtidos por Hasdorff, nestas mesmas condições, estão mostrados nas figuras 4.4a e 4.5a.

Analisando as figuras 4.4a e 4.4b, nota-se que a forma do controle gerado pelo GRAMIN, difere daquela obtida por Hasdorff . Esta diferença pode ser melhor observada comparando as duas últimas linhas da tabela 4.11, onde reproduzimos os valores numéricos do controle após 12 iterações. Este tipo de entrada gerada pelo GRAMIN, acarreta uma ligeira sobre-elevação na resposta do sistema (vide figura 4.5b).

Apesar de nossos resultados, não reproduzirem exatamente os de Hasdorff, a resposta do sistema (figura 4.5b) à nossa entrada de controle, é mais rápida, o custo de operação é menor (tabela 4.10), e a forma da entrada de controle se assemelha mais ao caso de funções contínuas. Esta semelhança de comportamento sugere que nosso resultado é coerente, pois o caso contínuo é o limite do amostrado, quando o número de intervalos de amostragem tendem para infinito.

Na tabela 4.11, tem-se os valores das entradas amostradas para o controle do motor DC, após 5 iterações com 4 reinicializações (IRREST=5) e para 12 iterações sem reinicialização. Todas as sequências de entradas de controle desta tabela, apresentam custo menor que o mostrado por Hasdorff. Os controles que mais se aproximam do seu resultado, são aqueles obtidos para 5 iterações com

TABELA 4.11 - VALORES DAS ENTRADAS AMOSTRADAS PARA O MOTOR DC.

NPTOS = 160 W_i = 1000 $u_0(k) = 0$; k = 0,1,...,7

NITER = 5	IRREST = 5								
IRREST	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	custo
1	3,59	2,69	1,81	0,93	-0,15	-0,98	-1,66	-0,98	338,12
2	4,18	2,74	1,55	0,55	-0,32	-1,08	-1,46	-0,45	315,89
3	4,45	2,73	1,40	0,37	-0,44	-1,05	-1,27	-0,19	302,67
4	4,57	2,70	1,30	0,27	-0,48	-1,00	-1,11	-0,01	298,71
5	4,72	2,74	1,26	0,20	-0,56	-1,08	-1,17	-0,03	296,01
NITER = 12		IRREST = 1							
Hasdorff	4,31	2,81	1,56	0,63	-0,31	-1,03	-1,44	-0,31	449,40
GRAMIN	6,94	2,91	0,24	-1,22	-1,70	-1,40	-0,36	1,87	275,20

IRREST=1 e IRREST=2. Entretanto, a sequência que apresenta menor custo, é aquela com 12 iterações sem reinicialização. Portanto, para este tipo particular de funcional de custo não se aplicam as sujestões de Fletcher e Reeves (44), sobre a reinicialização do método a cada (n+1) iterações.

4.4 - O CONTROLE DO MOTOR DC COM ENTRADAS MODULADAS POR LARGURA DE PULSO.

4.4.1 - PULSOS POSITIVOS E NEGATIVOS.

Neste caso, o motor é o mesmo da figura 4.1, porém vamos controlá-lo usando entradas moduladas por largura de pulso. A entrada de controle u(t) é dada pela equação (2.4.43), cuja forma está esquematizada na figura 4.5. Nota-se nesta figura, a limitação da amplitude dos pulsos em +1 ou -1. Dependendo da unidade utilizada para a medida da entrada de controle, tal valor de amplitude pode ser suficiente ou não. Sendo assim, consideramos que u(t) está na forma normalizada, e a tensão de campo e_f é dada por :

$$e_f(t) = Au(t) \qquad (4.4.1)$$

onde A é uma constante que define a amplitude dos pulsos.

Novamente, o problema é encontrar uma entrada que produza uma mudança em degrau de 10 radianos na posição do eixo do motor. São mantidas as equações de estado (4.1.11 - 4.1.14), o funcional de custo (4.1.9), as equações adjuntas (4.2.2 - 4.2.5) e sua condição terminal (4.2.6). Entretanto nestas equações, a tensão de campo é dada por (4.4.1). Ao se fazer tal substituição, o termo $R(e_f)^2$ adicionado ao funcional de custo, para limitar a energia do controle, varia com o quadrado de A. Para valores de A maiores que a unidade, o fun

cional de custo passa a depender predominantemente deste termo, induzindo larguras de pulso muito pequenas e invibializando a operação do sistema. Este problema pode ser contornado, substituindo nas equações do sistema R por R', onde:

$$R' = R/A^2 \qquad (4.4.2)$$

torna o termo de penalização da energia,independente de A. Sob tais considerações a equação (2.4.54), para a k-ésima componente do gradiente fica:

$$g_k = \left[0{,}1\lambda_3(t)A\,\mathrm{sgn}\tau_k + \lambda_4(T_k)R'A^2\right]\mathrm{sgn}\tau_k \cdot T \qquad (4.4.3)$$

No apêndice A5 estão listados o programa principal e as sub-rotinas desenvolvidas para a resolução deste caso.

RESULTADOS

Os resultados de Hasdorff foram obtidos para dois valores das funções de ponderação dos termos de penalização : $W_1=W_2=W_3=0$ e 100. Com $W_i=0$ foram feitas 5 iterações e para $W_i=100$ foram realizadas 10 iterações. O controle foi calculado no intervalo $[t_0=0;t_f=3\text{seg}]$, com intervalo de amostragem de 0,3 segundos de modo que M=10,e com largura inicial dos pulsos $\tau_0(i)=0$ para i=0,1,...,M-1 . Quanto a amplitude máxima da entrada de controle, o autor não fornece o valor utilizado.

Antes de comparar os resultados com os de Hasdorff, vamos escolher o melhor tipo de busca unidimensional a ser empregado , bem como o valor para a amplitude A da entrada de controle. Posteriormente será feita uma análise do algorítmo de Pagurek, pois neste espaço pode ocorrer saturação na largura dos pulsos($\tau_i \geq 1$).

1 - Análise das buscas lineares

Na tabela 4.12 mostramos o estado final do sistema e o custo, para os três tipos de busca linear, usando W_i=0 e 100. Para a amplitude da entrada de controle A, utilizamos os seguintes valores: 10 , 20, 30 e 100. As larguras de pulso encontram-se na tabela 4.14, sendo identificadas pelo número correspondente na coluna τ_i das tabelas 4.14.a e 4.14.b.

Nestes dados, com W_i=0, A igual a 10 e 100, as três buscas conduzem praticamente aos mesmos valores de custo. Entretanto com A igual a 20 e 30, a busca completa (c) leva a um valor de custo maior que as demais. No caso da amplitude A=30, com a busca (c), o estado final do sistema diverge dos valores desejados. Isto ocorre, porque a busca (c) divide o passo inicial por 2^{10}, ficando muito sensível à pequenas oscilações no valor do funcional de custo, que são interpretadas como mínimos locais pelo algorítmo. Observa-se entretanto , que o mesmo não ocorre com W_i=100 e A=30, onde a busca (c) apresenta os mesmos resultados da busca (b), pois a adição dos termos de penalização mascara as pequenas flutuações no funcional. A busca incompleta (a), com W_i=100 e A=30, apresenta um valor do funcional de custo 10% maior que o obtido com as demais buscas neste caso. Este fato pode ser explicado, porque esta busca faz a interpolação sem refinar o intervalo no qual o mínimo se localiza. Isto significa que o funcional de custo, nesta região, não apresenta o comportamento de uma curva quadrática, e o refinamento do intervalo de interpolação torna-se relevante.

Em média, em todos os casos analisados, os melhores resultados foram obtidos com as buscas incompletas (a) e (b), sendo os resultados da busca (b) ligeiramente melhores que os da busca (a). Nas próximas análises utilizaremos a busca (b), apesar de mais lenta que

TABELA 4. 12 - ANÁLISE DO DESEMPENHO DOS DIFERENTES TIPOS DE BUSCAS : ESPAÇO M.L.P.

AMOSTRAS DAS VARIÁVEIS DE ESTADO E DO CUSTO EM t = 3 seg.

NITER = 10 IRREST = 1 NPTOS = 180 IPGK = 0 $\tau_0(i) = 0$; i = 0,1,...,9

	A = 10						A = 20				
W_i	BUSCA	x_1	x_2	x_3	J(u)	τ_i	x_1	x_2	x_3	J(u)	τ_i
0	(a)	8,82672	5,32830	0,31256	190,564	1	12,03036	4,92669	0,04598	147,408	7
0	(b)	8,83966	5,30647	0,29939	190,571	2	12,58853	5,58985	0,14280	147,053	8
0	(c)	9,17414	5,85089	0,36821	190,984	3	19,05872	13,39998	0,84942	170,307	9
100	(a)	7,37016	2,50349	-0,45871	1537,848	4	9,44235	0,61255	-1,08533	340,667	10
100	(b)	7,32883	2,45646	-0,46321	1536,917	5	9,56835	0,41917	-0,95506	280,490	11
100	(c)	6,23051	1,48538	-0,49571	1868,501	6	9,64633	0,51249	-0,94559	281,098	12
	A = 30						A = 100				
W_i	BUSCA	x_1	x_2	x_3	J(u)	τ_i	x_1	x_2	x_3	J(u)	τ_i
0	(a)	12,05517	2,29361	-0,31240	129,538	13	5,81675	3,91621	0,32897	212,216	19
0	(b)	11,43940	1,35152	-0,43293	129,566	14	5,81675	3,91621	0,32897	212,216	20
0	(c)	1,74503	1,17486	0,09869	271,352	15	5,81675	3,91621	0,32897	212,216	21
100	(a)	9,77672	0,02778	-1,05734	256,106	16	9,83477	0,04499	-0,83132	196,800	22
100	(b)	9,84468	0,16850	-0,94001	230,170	17	9,75909	1,10912	-0,70576	314,527	23
100	(c)	9,84468	0,16850	-0,94001	230,170	18	5,31130	2,60670	-0,04209	3091,634	24

a (a), vide tabela 4.6, pois estamos interessados na qualidade dos resultados à despeito do tempo de computação.

2- Resposta do sistema a diferentes amplitudes do controle.

Como Hasdorff em seu trabalho (28), não cita qual a amplitude dos pulsos utilizada em seu exemplo, na tabela 4.13 apresentamos os resultados obtidos para o estado final do sistema e para o funcional de custo, com diferentes amplitudes A do controle (vide equação 4.4.1). As larguras de pulsos correspondentes estão nas tabelas 4.14.a e 4.14.b .

Com A=1, tanto para W_i=0 como W_i=100, a energia aplicada não é suficiente para levar o controle ao estado final desejado. Com A=100 e W_i=0, a tensão de campo é alta, a largura dos pulsos torna-se muito estreita e o controle perde a resolução (vide a linha 20 da tabela 4.14.b). Entretanto neste caso com W_i=100, apesar do estreitamento dos pulsos, ainda é possível controlar o sistema.

O menor custo e a melhor resposta do sistema são obtidos com A=30, tanto para W_i=0 como W_i=100. Com este valor de A, o controle é composto de pulsos cuja magnitude é 30 volts, com larguras de pulsos definidas nas linhas 14 e 17 da tabela 4.14.b. Adotaremos esta amplitude na comparação de nossos resultados com os de Hasdorff, uma vez que ele não menciona qual valor utilizou em seu exemplo.

3 - Análise comparativa.

Na figura 4.6a e 4.7a estão os resultados obtidos por Hasdorff respectivamente para a resposta do sistema e a largura dos pulsos. Na figura 4.6a, a curva 1 foi construída com W_i=0 e 5 iterações, e a curva 2 com W_i=100 e 10 iterações. As larguras de pulso que definem

TABELA 4.13 - DIFERENTES TENSÕES DE ENTRADA : M.L.P.

RESPOSTA DO SISTEMA EM t = 3 seg.

NITER = 10 IRREST = 1 IPGK = 0 NPTOS = 180

$\tau_0(i) = 0$; $i = 1,2,\ldots,9$

$W_i = 0$					
A	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	J(u)	τ_i
1	0,73824	0,39190	0,02074	294,476	25
10	8,83966	5,30647	0,29939	190,571	2
20	12,58853	5,58985	0,14280	147,053	8
30	11,43940	1,35152	-0,43293	129,566	14
100	5,81675	3,91621	0,32897	212,216	20
$W_i = 100$					
1	0,95708	0,70442	0,07698	8526,816	26
10	7,32883	2,45646	-0,46321	1536,917	5
20	9,56835	0,41917	-0,95506	280,490	11
30	9,84468	0,16850	-0,94001	230,170	17
100	9,75909	1,10912	-0,70576	314,527	23

BUSCA INCOMPLETA B

TABELA 4.14.a - VALORES DAS ENTRADAS DE CONTROLE PARA AS TABELAS 4.12 e 4.13.

(LARGURA DOS PULSOS τ_i) : CASO M.L.P.

τ_i	τ_0	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9
1	1,00000	1,00000	1,00000	1,00000	1,00000	0,90539	0,37664	0,00093	0,02441	0,02002
2	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	0,36115	0,02104	-0,05233	0,01443
3	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	0,97196	0,15562	0,00078	-0,04419
4	1,00000	1,00000	1,00000	1,00000	1,00000	0,65037	-1,00000	-1,00000	-1,00000	-1,00000
5	1,00000	1,00000	1,00000	1,00000	1,00000	0,60141	-1,00000	-1,00000	-1,00000	-1,00000
6	1,00000	1,00000	1,00000	1,00000	0,41000	-0,14655	-0,67083	-1,00000	-1,00000	-1,00000
7	1,00000	1,00000	1,00000	0,56297	0,05148	-0,12686	-0,16537	-0,16593	-0,10568	-0,00444
8	1,00000	1,00000	1,00000	0,67000	0,04373	-0,06142	-0,08135	-0,07447	-0,00208	-0,05425
9	0,99913	0,99587	0,99390	0,99344	0,99412	0,99547	0,99698	0,99821	0,99887	-0,99903
10	1,00000	1,00000	0,93536	0,45728	-0,00015	-0,42119	-0,76334	-0,98088	-0,99981	-0,75439
11	1,00000	1,00000	1,00000	0,65086	0,00414	-0,80721	-1,00000	-1,00000	-0,99093	-0,38331
12	1,00000	1,00000	1,00000	0,62176	0,00434	-0,73652	-1,00000	-1,0000	-1,00000	-0,36171
13	1,00000	1,00000	1,00000	0,02859	-0,52064	-0,57094	-0,49124	-0,26465	-0,03383	-0,05078

TABELA 4.14.b - VALORES DAS ENTRADAS DE CONTROLE PARA AS TABELAS 4.12 E 4.13.

(LARGURA DOS PULSOS τ_i) : CASO M.L.P.

τ_i	τ_0	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9
14	1,00000	1,00000	1,00000	0,06707	-0,59027	-0,74493	-0,62261	-0,33153	-0,03891	-0,01551
15	0,00008	0,00006	0,00005	0,00003	0,00002	0,00001	0,00001	0,00000	0,00000	0,00000
16	0,72703	0,83642	0,58591	0,15823	-0,17576	-0,37693	-0,55475	-0,59307	-0,61848	-0,49034
17	0,99609	0,71890	0,42954	0,15833	-0,09968	-0,34160	-0,51645	-0,60990	-0,57675	-0.36551
18	0,99049	0,71713	0,43103	0,16247	-0,09366	-0,33443	-0,51130	-0,60968	-0,58545	-0,38560
19	0,00067	0,00052	0,00038	0,00027	0,00018	0,00011	0,00006	0,00003	0,00001	0000000
20	0,00048	0,00037	0,00027	0,00019	0,00013	0,00008	0,00004	0,00002	0,00000	0,00000
21	0,00036	0,00028	0,00021	0,00015	0,00010	0,00006	0,00003	0,00001	0,00000	0,00000
22	0,24514	0,18084	0,11852	0,05359	-0,00872	-0,06518	-0,11112	-0.14004	-0,14271	-0,10637
23	0,22133	0,16550	0,11124	0,05478	0,00047	-0,04888	-0,08935	-0,11513	-0,11836	-0,08819
24	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	-0,00000	-0,00000	-0,00000
25	1,00000	1,00000	1,00000	0,93431	0,18154	0,38526	0,05541	0,05362	0,11011	0,10470
26	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	0,38527

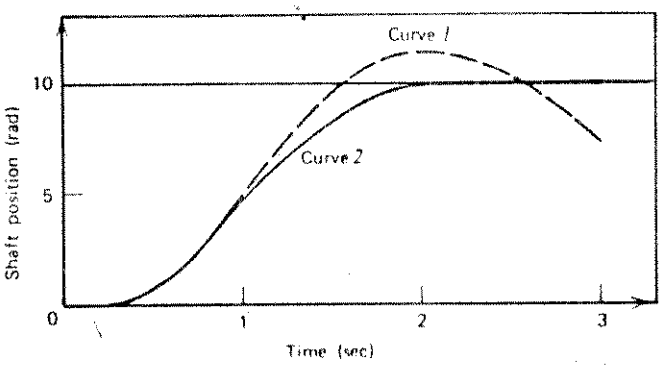


Fig. 4.6a - Resposta do motor DC com entradas moduladas por largura de pulso , obtida por Hasdorff.

caso MLP: PULSOS POSITIVOS E NEGATIVOS.

curva 1: $W_i = 0$; 5 iterações.

curva 2: $W_i = 100$; 10 iterações.

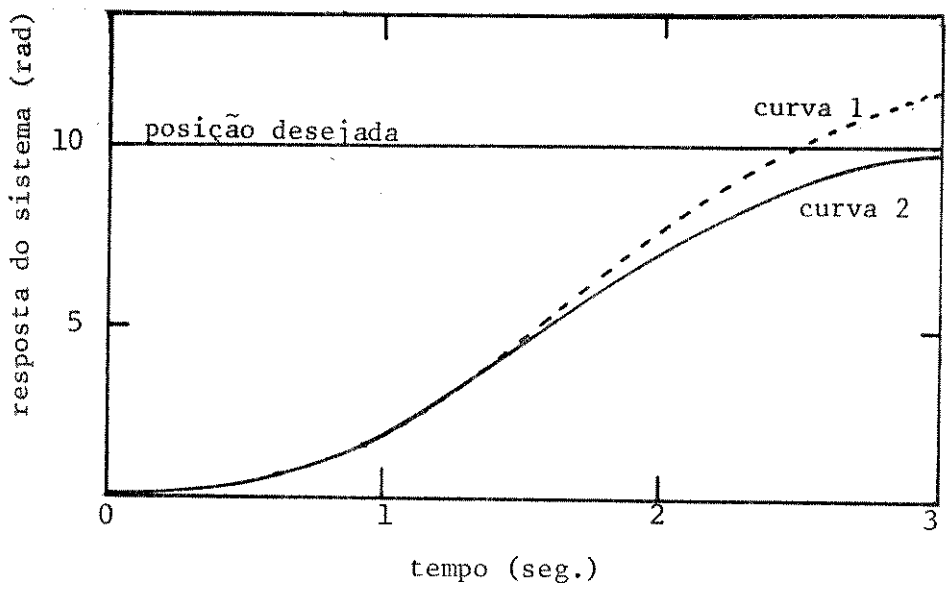


Fig. 4.6b - Resposta do motor DC, com entradas moduladas por largura de pulso:PULSOS POSITIVOS E NEGATIVOS: GRAMIN.

NPTOS = 180 IRREST = 1

IPGK = 0 ; busca b.

tensão de entrada = 30 V.

curva 1 : $W_i = 0$;NITER = 5

curva 2 : W_i=100;NITER = 10

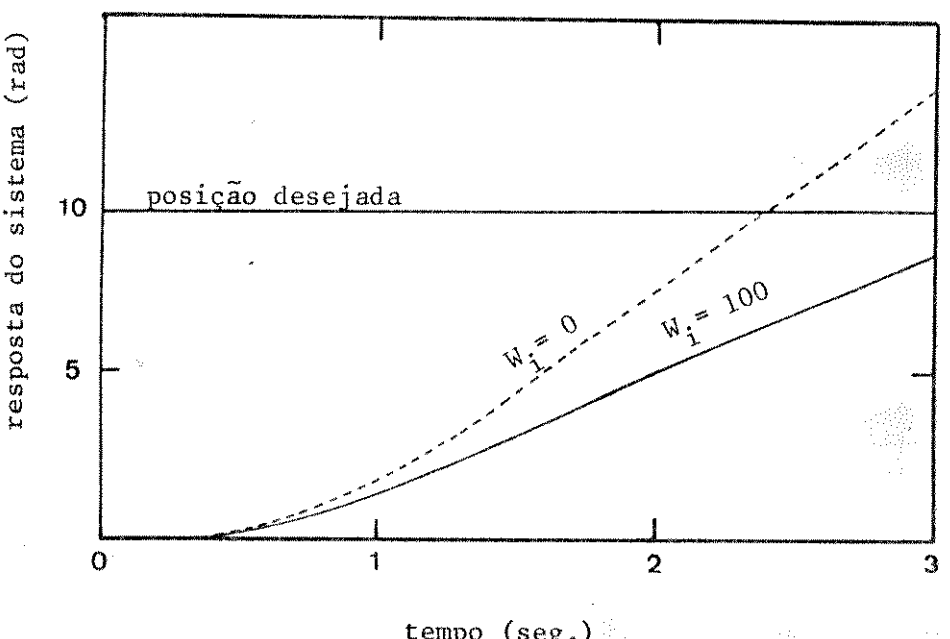


Fig. 4.8 - Resposta do motor DC, com entradas moduladas por largura de pulso: PULSOS POSITIVOS.

NPTOS = 180 IRREST = 1

IPGK = 0 NITER = 5

tensão de entrada = 30 V.

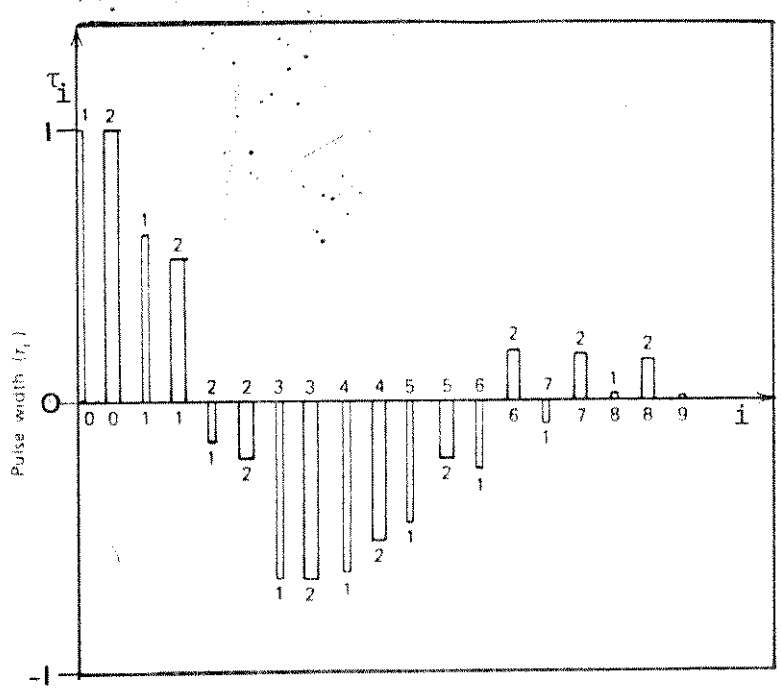


Fig. 4.7a - Largura das entradas de controle obtidas por Hasdorff, para o motor DC. caso MLP: PULSOS POSITIVOS E NEGATIVOS.

1 → W_i = 0; 5 iterações
2 → W_i = 100; 10 iterações

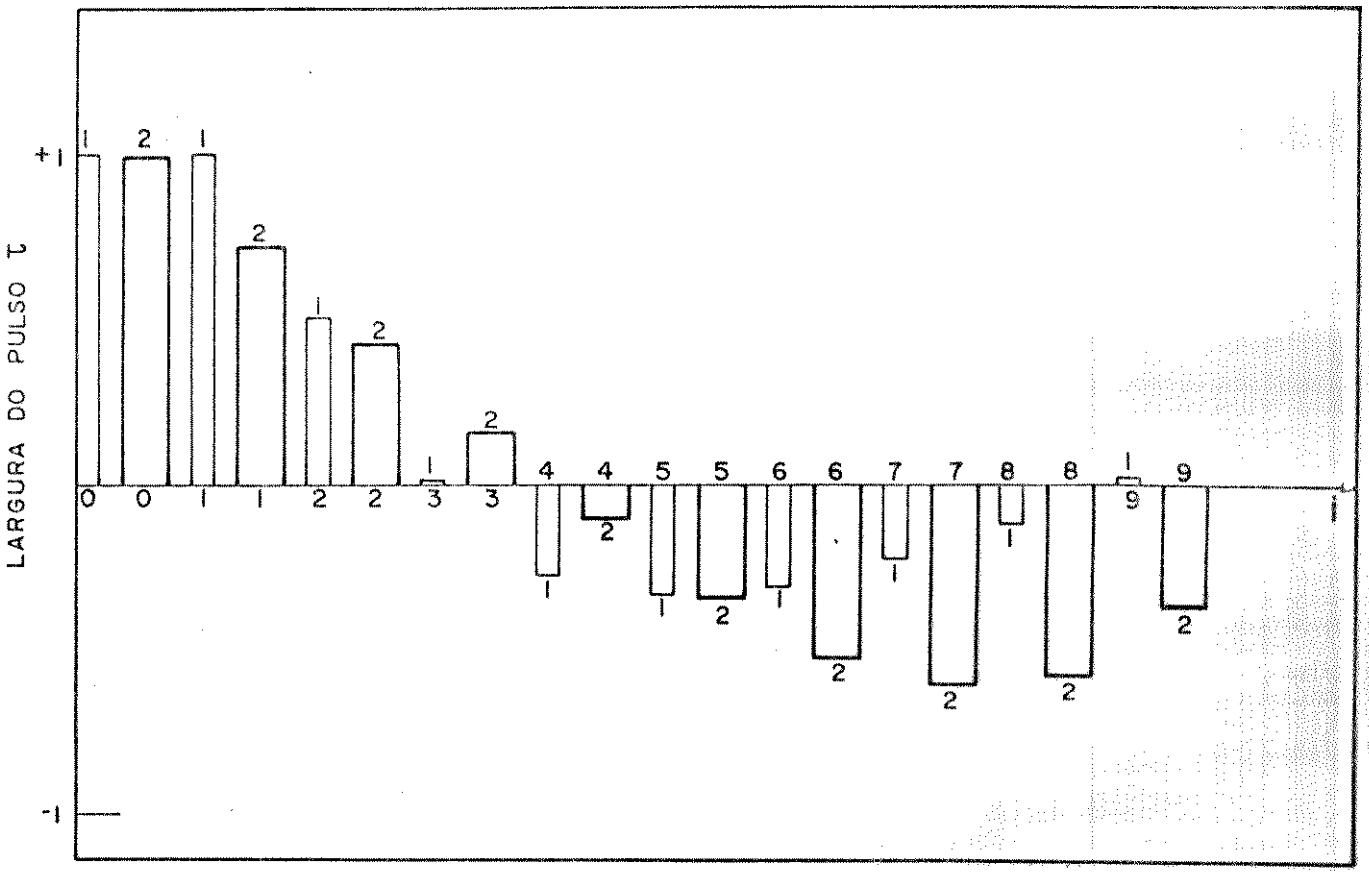


Fig. 4.7b - Largura das entradas de controle para o motor DC, obtidas com o GRAMIN, para o caso de modulação por largura de pulsos: PULSOS POSITIVOS E NEGATIVOS. NPTOS = 180 IRREST = 1 IPGK = 0

1 → W_i = 0; 5 iterações
2 → W_i = 100; 10 iterações

tensão de entrada = 30 volts.
$\tau_0(i) = 0$; i = 1,2,...,9

a entrada de controle para as curvas 1 e 2 estão mostradas na figura 4.7a e identificadas pelos números 1 e 2.

Para a comparação dos resultados, usamos A=30, e a execução do GRAMIN foi realizada com a busca incompleta (b). Os nossos resultados estão mostrados nas figuras 4.6b e 4.7b, onde as curvas 1 e 2 foram construídas sob as mesmas condições daquelas de Hasdorff.

Comparando as curvas 1, das figuras 4.6a e 4.6b, e respectivas entradas de controle, observamos que os resultados diferem, e ambos não mostram uma resposta satisfatória. A resposta obtida por Hasdorff apresenta um comportamento oscilatório, ao passo que a respota dada pelo GRAMIN, ultrapassa o valor pré-estabelecido para o estado final. Isto ocorre porque não há penalização no custo, para forçar o estado final desejado.

Nas curvas 2, onde são considerados os termos de penalização, o estado final é praticamente atingido, porém a resposta obtida por Hasdorff é mais rápida, pois seu sistema atinge o estado final em t=2 segundos. Contudo esta diferença de resultados, pode estar no fato que desconhecemos a tensão de campo empregada por Hasdorff.

4 - Aplicação do algorítmo de Pagurek

O algorítmo de Pagurek (49), pode ser usado quando ocorre saturação na variável de controle. No programa GRAMIN, a variável IPGK define o uso do algorítmo de Pagurek, se IPGK=1 este algorítmo é executado.

Os resultados da aplicação do método, estão mostrados na tabela 4.15, para amplitudes do controle iguais a 10 e 30 e W_i=100. O valor A=10 foi escolhido, porque ele causa saturação no controle qualquer que seja W_i. Para efeito de comparação com os casos em que não há saturação, usamos A=30 e W_i=100.

TABELA 4.15 - APLICAÇÃO DO ALGORÍTMO DE PAGUREK

ESPAÇO M.L.P.

ESTADO FINAL DO SISTEMA E CUSTO							
IPGK	A	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	J(u) ~~	τ_i	NITER = 10 IRREST = 1 NPTOS = 180 W_i = 100 $\tau_0(i) = 0$; i = 1,2,...,9
0	10	7,32883	2,45646	-0,46321	1536,917	(a)	
1	10	7,32886	2,45851	-0,44696	1536,346	(b)	
0	30	9,84468	0,16850	-0,94001	230,170	(c)	
1	30	9,84468	0,16850	-0,94001	320,170	(d)	

CONTROLE OBTIDO (LARGURA DOS PULSOS)										
τ_i	τ_0	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9
(a)	1,00000	1,00000	1,00000	1,00000	1,00000	0,60141	-1,00000	-1,00000	-1,00000	-1,00000
(b)	1,00000	1,00000	1,00000	1,00000	1,00000	0,59768	-1,00000	-1,00000	-1,00000	-0,94440
(c)	0,99609	0,71890	0,42954	0,15833	-0,09968	-0,34160	-0,51645	-0,60990	-0,57675	-0,36551
(d)	0,99609	0,71890	0,42954	0,15833	-0,09968	-0,34160	-0,51645	-0,60990	-0,57675	-0,36551

TABELA 4.16 - ESTUDO DAS DIFERENTES REINICIALIZAÇÕES: CASO M.L.P.

NITER = 5 IRREST = 5 NPTOS = 180 W_i = 100 $\tau_0(i) = 0$; i = 1,2,...,9

BUSCA B

A = 10 IPGK = 0						A = 10 IPGK = 1				
IRREST	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	J(u)	τ_i	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	J(u)	τ_i
1	7,32883	2,45646	-0,46321	1536,917	I	7,32886	2,45851	-0,44696	1536,346	V
2	7,32886	2,45851	-0,44696	1536,346	II	7,32886	2,45851	-0,44696	1536,346	VI
3	7,32886	2,45851	-0,44696	1536,346	III	7,32886	2,45851	-0,44696	1536,346	VII
4	7,32886	2,45851	-0,44696	1536,346	IV	7,32886	2,45851	-0,44696	1536,346	**
5	7,32886	2,45851	-0,44696	1536,346	*	7,32886	2,45851	-0,44696	1536,346	**

LARGURAS DOS PULSOS

τ_i	τ_0	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9
I	1,00000	1,00000	1,00000	1,00000	1,00000	0,56863	-1,00000	-1,00000	-1,00000	-1,00000
II	1,00000	1,00000	1,00000	1,00000	1,00000	0,60870	-1,00000	-1,00000	-1,00000	-0,94145
III	1,00000	1,00000	1,00000	1,00000	1,00000	0,61101	-1,00000	-1,00000	-1,00000	-0,94430
IV	1,00000	1,00000	1,00000	1,00000	1,00000	0,61111	-1,00000	-1,00000	-1,00000	-0,94442
V	1,00000	1,00000	1,00000	1,00000	1,00000	0,59749	-1,00000	-1,00000	-1,00000	-0,94416
VI	1,00000	1,00000	1,00000	1,00000	1,00000	0,59771	-1,00000	-1,00000	-1,00000	-0,94444
VII	1,00000	1,00000	1,00000	1,00000	1,00000	0,59772	-1,00000	-1,00000	-1,00000	-0,94444

Quando A=10 verifica-se da tabela 4.15, que o custo obtido com o uso do algorítmo cai de 1536,917 para 1536,346. A diferença percentual neste caso é 0,034%. Resultados semelhantes são encontrados à partir dos valores de custo obtidos por Pagurek e Woodside, no exemplo I da referência 49, onde a diferença percentual é 0,075%.

Com A=30, nota-se que é indiferente o uso do algorítmo de Pagurek, pois não há saturação e os resultados obtidos com IPGK=0 e IPGK=1, são idênticos.

Na tabela 4.16, mostramos o comportamento do algorítmo, com menos iterações e variando o número de reinicializações. Com IPGK=0 e NITER=5, o custo se estabiliza em 1536,346, somente após uma reinicialização. Porém com IPGK=1, este valor é atingido ao cabo de 5 iterações, sem reinicialização (IRREST=1). Portanto, o uso do algorítmo de Pagurek aumentou a velocidade de convergência do método.

4.4.2 - PULSOS POSITIVOS.

O controle do motor DC será feito, considerando uma entrada de controle constituída somente por pulsos positivos (figura 2.6). Este tipo de entrada foi discutida na parte B da seção 2.4.5. Quanto ao modelamento do sistema, todas as equações do caso anterior são mantidas, com exceção da entrada de controle, dada pela equação (2.4.55), e a expressão para o gradiente, que tem a forma:

$$g_k = \left[0{,}1\lambda_3 A + R' A^2 \lambda_4 \right] T \qquad (4.4.4)$$

A tabela 4.17 mostra o estado final do sistema (t_f=3seg) e o custo, obtidos após 5 iterações e uma reinicialização, para diferentes valores de amplitude dos pulsos, com e sem penalização do funcional de custo. As larguras de pulsos correspondentes, também

TABELA 4.17 - DIFERENTES TENSÕES DE ENTRADA ; CASO M.L.P (PULSOS POSITIVOS)

NITER = 5 IRREST = 2 IPGK = 0 NPTOS = 180 $\tau_0(i) = 0$; i = 1,2,...,9

	$W_i = 0$					$W_i = 100$				
A	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	J(u)	τ_i	x_1 (rad)	x_2 (rad/seg)	x_3 (A)	J(u)	τ_i
10	8,72941	5,14223	0,27300	190,322	I	7,14676	3,53248	0,13372	2257,972	IV
20	13,01240	6,10579	0,21135	146,599	II	8,40578	3,40069	0,09281	1574,149	V
30	13,67763	5,64610	0,15894	131,354	III	8,58526	3,23988	0,07967	1403,542	VI

CONTROLE OBTIDO (LARGURA DOS PULSOS)

τ_i	τ_0	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9
I	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	0,22222	0,00000	0,00000	0,00000
II	1,00000	1,00000	1,00000	0,72921	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
III	1,00000	1,00000	0,38572	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
IV	1,00000	1,00000	1,00000	1,00000	0,33321	0,00000	0,00000	0,00000	0,00000	0,00000
V	1,00000	1,00000	0,11147	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
VI	1,00000	0,38811	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000

estão incluídos nesta tabela.

Dentre os 3 valores de A considerados, a melhor resposta do sistema, segundo o critério de menor custo, é obtida com A=30, a exemplo do caso anterior. Com esta amplitude de pulso, a posição do eixo do motor em função do tempo ($x_1(t)$), está mostrada na figura 4.8 . Para $W_i=0$, pode-se notar que o motor ultrapassa a posição desejada, e em $t=t_f$ sua velocidade angular ainda não é nula, sendo igual a 5,65 rad/seg. Com $W_i=100$, a resposta obtida é melhor, mas o sistema não atinge a posição final desejada e a velocidade angular ainda está em 3,24 rad/seg. Este tipo de comportamento, apresentado pela resposta do sistema, é físicamente coerente levando-se em conta que no modelo não há inversão da tensão de campo, para frear o motor. Neste caso, a única ação de frenagem é exercida pelas forças dissipativas, tais como atrito dinâmico e resistência de campo. Note o comportamento das entradas de controle, que atuam nos instantes iniciais e depois permanecem desligadas (tabela 4.17: linhas III e VI).

Este controle com modulação por largura de pulsos, considerando-se somente pulsos positivos, apresentaria um melhor desempenho, quando aplicado a um sistema em que o efeito das forças dissipativas fosse maior.

4.5 - DETERMINAÇÃO DE PARÂMETROS DE UM COMPENSADOR PÓLO ZERO.

O problema a ser resolvido é a escolha dos parâmetros ótimos de um compensador pólo zero para um servo-posicionador, com o objetivo de melhorar a resposta do sistema. O servo-posicionador está esquematizado na figura 4.9 .

4.5.1 - MODELAMENTO DO SISTEMA.

O modelo do sistema será desenvolvido adotando-se os mesmos

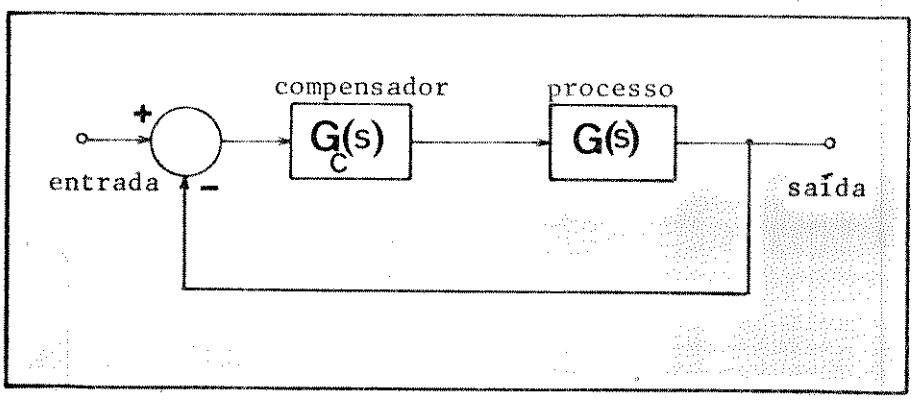


Fig. 4.9 - Diagrama esquemático do servo-posicionador.

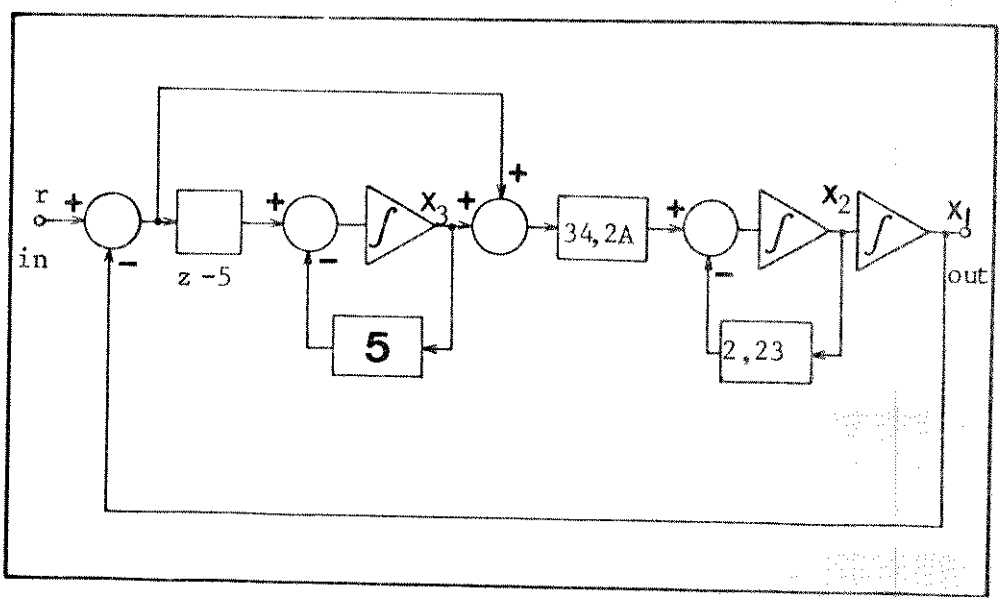


Fig 4.10 - Diagrama de simulação para o servo-posicionador da figura 4.9 .

parâmetros usados por Hasdorff (28), para permitir a comparação dos resultados.

A função de transferência do processo é:

$$G(s) = \frac{34,2}{s(s + 2,23)(s + 5)} \qquad (4.5.1)$$

E a equação para o compensador pólo zero é dada por:

$$G_c(s) = \frac{A(s + z)}{1 + s/p} \qquad (4.5.2)$$

A resposta do sistema é acelerada se é utilizado um compensador avanço (1,3), e na equação (4.5.2) deve-se ter $z < p$. Se o valor do pólo do compensador for muito maior que o valor dos pólos do processo, sua resposta transiente pode ser desprezada e a equação (4.5.2) pode ser simplificada como:

$$G_c(s) = A(s + z) \qquad (4.5.3)$$

Da figura 4.9 e das equações (4.5.1) e (4.5.3), pode-se construir o diagrama de simulação para o servo-posicionador, mostrado na figura 4.10. Deste diagrama as equações de estado são diretamente obtidas:

$$\dot{x}_1 = x_2 \qquad (4.5.4)$$

$$\dot{x}_2 = -34,2Ax_1 - 2,23x_2 + 34,2Ax_3 + 34,2A\,r \qquad (4.5.5)$$

$$\dot{x}_3 = -(z - 5)x_1 - 5x_3 + (z - 5)r \qquad (4.5.6)$$

Onde A e z são os parâmetros que devem ser estimados, para otimizar o desempenho do servo-posicionador. A entrada r(t), que deve ser seguida pela saída x_1, é especificada como a função degrau

unitário, pois se o sistema apresentar uma boa resposta ao degrau, o mesmo sucederá com qualquer outra função. Neste caso:

$$r(t) = \underline{1}(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \qquad (4.5.7)$$

O funcional de custo é definido de forma a minimizar o erro de posição durante a operação do sistema, no intervalo de tempo $[t_0, t_f]$.

$$\text{erro} = r - x_1 \qquad (4.5.8)$$

O intervalo de tempo é escolhido, considerando-se que o sistema é invariante no tempo, e que o pólo dominante implica em uma constante de tempo de 0,45 segundos. Desta forma, pode-se escolher t_0=0 e t_f=2seg, onde 2 segundos corresponde aproximadamente a quatro constantes de tempo, que é um intervalo razoável para o sistema entrar em regime permanente.

Com estas considerações o funcional de custo tem a forma:

$$J(\underline{u}) = \int_{t=0}^{2seg.} (1 - x_1)^2 \, dt + W_1(1 - x_1)^2 + W_2 x_2^2 \Big|_{t=2seg} \qquad (4.5.9)$$

Definindo $\dot{x}_4 = (1 - x_1)^2$, $x_4(0) = 0$ (4.5.10)

tem-se:

$$J(\underline{u}) = \phi(x(t_f)) = x_4 + W_1(1 - x_1)^2 + W_2 x_2^2 \qquad (4.5.11)$$

Os termos de penalização foram adicionados, para assegurar o estado final do sistema, ou seja:

$$x_1(t_f) = 1 \qquad (4.5.12)$$

$$x_2(t_f) = 0 \tag{4.5.13}$$

As equações de estado do sistema são constituídas pelas equações (4.5.4), (4.5.5), (4.4.6) e (4.5.10), sujeito à condição inicial:

$$\underline{x}(0) = C = 0 \tag{4.5.14}$$

A entrada de controle para minimizar o critério de custo é:

$$\underline{u} = \begin{bmatrix} A \\ z \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.5.15}$$

Para este sistema a equação adjunta fica:

$$\dot{\lambda}_1 = -\lambda_2 \tag{4.5.16}$$

$$\dot{\lambda}_2 = 34{,}2\,A\,\lambda_1 + 2{,}23\lambda_2 - 34{,}2\,A\,\lambda_3 \tag{4.5.17}$$

$$\dot{\lambda}_3 = (z - 5)\lambda_1 + 5\lambda_3 \tag{4.5.18}$$

$$\dot{\lambda}_4 = 2(1 - x_1)\lambda_1 \tag{4.5.19}$$

Sujeitas à condição final:

$$\underline{\lambda}(t_f) = \nabla_x \phi(\underline{x}(t_f,u)) = \left[2W_1(x_1 - 1) \quad 2W_2x_2 \quad 0 \quad 1 \right]_{t=2} \tag{4.5.20}$$

A expressão para o gradiente é obtida considerando que:

$$f_u(\underline{x},\underline{u}) = \begin{bmatrix} 0 & 0 \\ 34{,}2(x_3 - x_1 + 1) & 0 \\ 0 & (1-x_1) \\ 0 & 0 \end{bmatrix} \tag{4.5.21}$$

Então o gradiente é dado por:

$$g\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \int_0^2 \{ 34{,}2 \left[x_3(t) - x_1(t) + 1 \right] \lambda_2(t) \} \, dt \\ \int_0^2 \{ \left[1 - x_1(t) \right] \lambda_3(t) \} \, dt \end{bmatrix} \quad (4.5.22)$$

No apêndice A6 está a listagem do programa para a resolução deste exemplo.

4.5.2 - RESULTADOS.

Os valores iniciais da entrada de controle u_0, são os mesmos utilizados por Hasdorff. Eles foram escolhidos de forma a satisfazer os seguintes requisitos: o parâmetro $z^0 = u_2^0 = 2{,}23$ foi ajustado neste valor para cancelar o pólo dominante em -2,23; a constante de ganho $A = u_1^0 = -0{,}365$ foi escolhida para que o par de pólos dominantes da malha fechada, tivessem um coeficiente de amortecimento de 0,7 , resultando em uma sobre-elevação de 6%. Com estes valores iniciais de u_0, o programa foi executado com 8 iterações, sem reinicialização do método e com constantes de ponderação iguais a 0 e 1000. Os resultados obtidos estão mostrados na tabela 4.18, onde também foram incluídos os resultados de Hasdorff com ponderação 1000. Comparando os dados para $W_i = 1000$, nota-se que o custo de Hasdorff se estabiliza à partir da quinta iteração, enquanto que no GRAMIN o custo já converge na 3º iteração, para um valor menor que o de Hasdorff. Quanto aos parâmetros A e z, a convergência de A é obtida na 4º iteração e a de z na 5º, sendo que os valores alcançados estão próximos aos de Hasdorff. Na figura 4.11a, tem-se os resultados de

TABELA 4.18 - VALORES OBTIDOS COM O GRAMIN COMPARADOS COM OS DE HASDORFF: CONJUNTO DE PARÂMETROS

NPTOS = 200 IRREST = 1 NITER = 8 $\underline{u}_0 = \begin{bmatrix} A^0 \\ z^0 \end{bmatrix} = \begin{bmatrix} 0,365 \\ 2,23 \end{bmatrix}$

	GRAMIN $W_i = 0$				GRAMIN $W_i = 1000$				HASDORFF $W_i = 1000$	
NITER	A	z	J(u)	$\|\|g\|\|^2$	A	z	J(u)	$\|\|g\|\|^2$	J(u)	$\|\|g\|\|^2$
1	8,2564	2,5933	0,1168	0,517E-3	0,4653	2,1812	0,2950	0,346E-2	1,370	0,135E-1
2	9,8483	-0,1294	0,0777	0,178E-4	0,4644	2,1790	0,2950	0,258E-1	0,296	0,223E 0
3	14,6907	-0,5407	0,0745	0,115E-3	0,4629	2,1325	0,2936	0,144E-1	0,295	0,247E-2
4	15,8246	-0,2753	0,0726	0,186E-4	0,4621	2,1308	0,2936	0,121E-3	0,295	0,122E 0
5	18,6554	-0,5568	0,0715	0,298E-4	0,4621	2,1307	0,2936	0,590E-4	0,294	0,395E-2
6	19,6777	-0,3815	0,0708	0,163E-4	0,4621	2,1307	0,2936	0,521E-4	0,294	0,129E-3
7	37,8349	-0,9246	0,0670	0,314E-4	0,4621	2,1307	0,2936	0,519E-4	0,294	0,646E-4
8	39,0680	-0,7348	0,0661	0,110E-4	0,4621	2,1307	0,2936	0,468E-4	0,294	0,659E-4
Δ	8,77 †	-0,0171 †	0,0792 †	*	0,464 ††	2,12 ††	0,294 ††	0,659E-4 ††		

Δ - VALOR OBTIDO POR HASDORFF: (†) após 5 iterações.
(††) após 8 iterações.

* - DADOS NÃO DISPONÍVEIS.

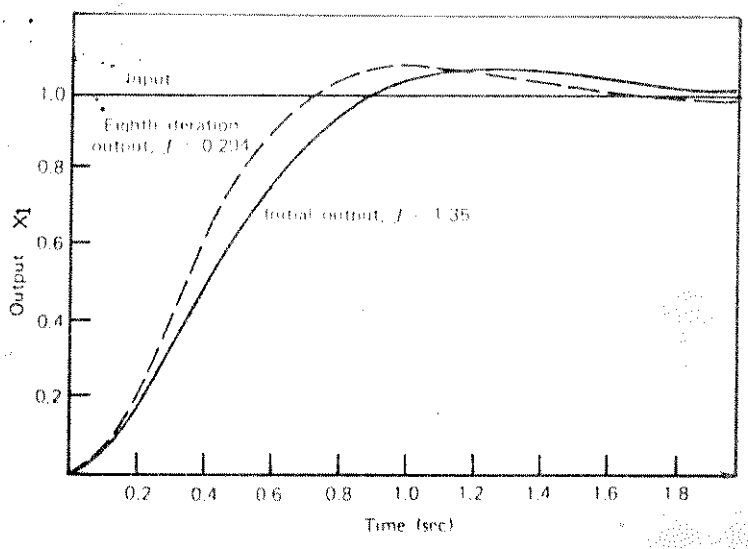


Fig. 4.11a - Resposta inicial e após 8 iterações obtida por Hasdorff , para o servo-posicionador. W_i = 1000.

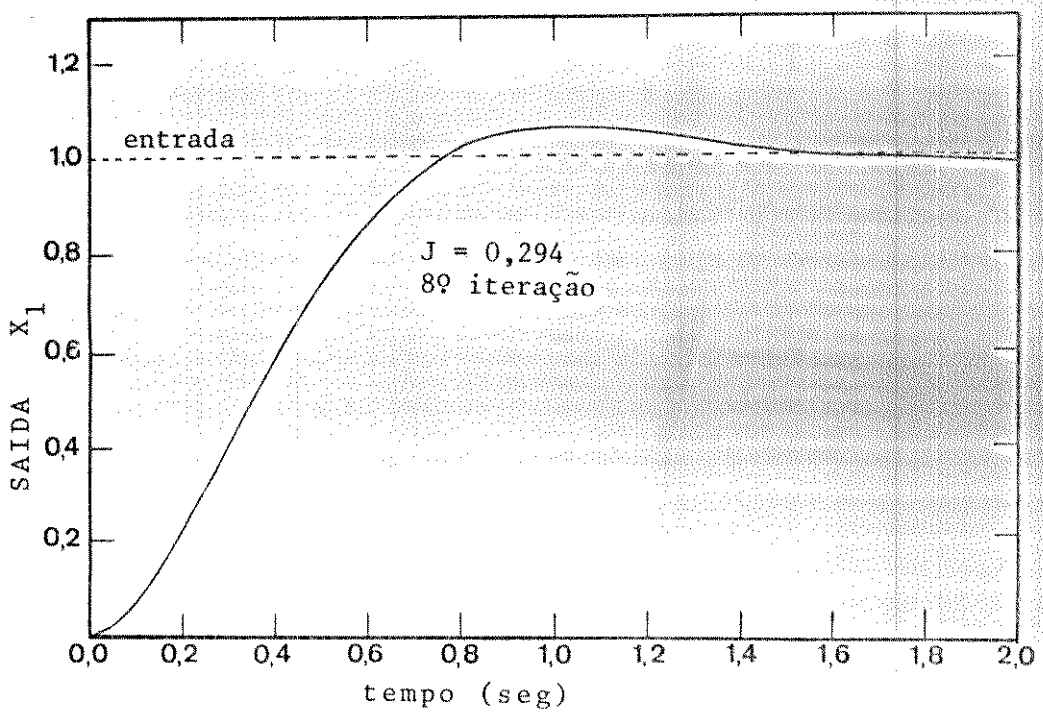


Fig. 4.11b - Resposta após 8 iterações para o servo-posicionador obtida com o GRAMIN. W_i = 1000.

u_1^0= -0,365 u_2^0= 2,23 NPTOS = 200 IRREST = 1

Hasdorff para W_i=1000. A curva contínua representa a resposta do sistema com o valor inicial u_0 estimado pela teoria clássica de compensação (1,3), e a curva tracejada fornece a resposta do sistema após a 8ª iteração. Na figura 4.11b, temos os resultados fornecidos pelo GRAMIN com 8 iterações. Note-se que a curva obtida com o GRAMIN mostra resultados melhores, pois seu tempo de resposta é menor.

Para $W_i = 0$, segundo Hasdorff, A e z podem ser determinados analiticamente pela teoria de filtragem de Wiener, e os valores obtidos são: $A \to \infty$ e $z = 0$, resultando em $J = 0$. Com 5 iterações e $W_i = 0$, Hasdorff obteve para o custo o valor $J = 0{,}0792$ com $A = 8{,}77$ e $z = -0{,}0171$. Neste caso, os resultados fornecidos pelo GRAMIN são: $J = 0{,}0715$, $A = 18{,}66$ e $z = -0{,}5568$; que estão mais próximos dos valores analíticos. Uma redução maior no custo é alcançada com 8 iterações e os valores obtidos são: $A = 39{,}07$, $z = -0{,}7348$ e $J = 0{,}0661$.

Na figura 4.12a estão os resultados de Hasdorff com 5 iterações e na figura 4.12b os resultados obtidos com o GRAMIN na 2ª iteração, ambos com $W_i = 0$. A segunda iteração foi escolhida, pois o custo já é menor que o fornecido por Hasdorff na 5ª iteração (vide tabela 4.18). Observamos também, que a medida que A, z e J vão tendendo para os valores analíticos, há um aumento na sobre-elevação e na frequência de oscilação, porém o tempo de resposta diminui. Com 2 iterações a sobre-elevação é 0,47 e o tempo de resposta é 1,3 segundos. Com 8 iterações a sobre-elevação é 0,68 e o tempo de resposta é 1,18 seg. Foi usado o critério de $\pm 1\%$ do valor em regime, para o cálculo do tempo de resposta.

Observamos na tabela 4.18, que as maiores variações nos resultados obtidos pelo GRAMIN ocorrem nas primeiras 5 iterações . Para verificar as sujestões de Fletcher e Reeves(44), a respeito

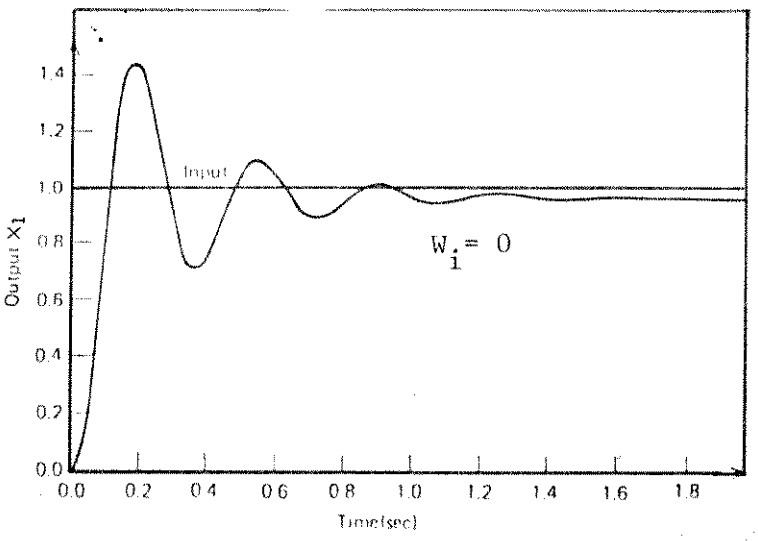


Fig. 4.12a - Resposta após 5 iterações obtida por Hasdorff para o servo-posicionador. $W_i = 0$.

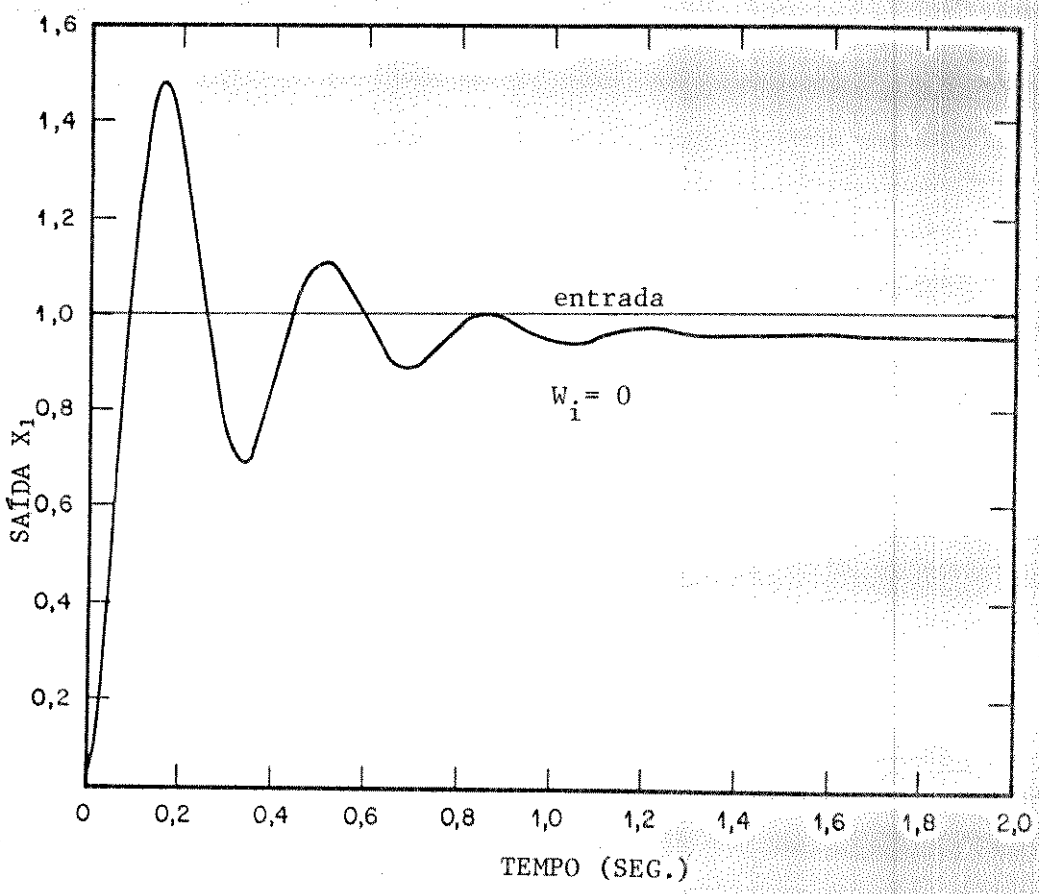


Fig 4.12b - Resposta após 2 iterações obtida com o GRAMIN para o servo-posicionador. $W_i = 0$

$u_1^0 = -0,365$ $u_2^0 = 2,23$ NPTOS = 200 IRREST = 1

TABELA 4.19 - INFLUÊNCIA DAS REINICIALIZAÇÕES: ENTRADAS NO ESPAÇO DE CONJUNTO DE PARÂMETROS.

NITER = 200 NITER = 5 IRREST = 4 $\underline{u}_0 = \begin{bmatrix} 0,365 \\ 2,23 \end{bmatrix}$

$W_i = 0$				
IRREST	A	z	J(u)	$\|\|g\|\|^2$
1	18,6554	-0,5568	0,0715	0,298E -4
2	1506,25	-1,0284	0,0105	0,390E -9
3	1506,13	-1,6180	0,0104	0,671E-10
4	1506,13	-1,6181	0,0104	0,670E-10
$W_i = 1000$				
1	0,4621	2,1307	0,2936	0,590E-4
2	0,4621	2,1307	0,2936	0,407E-4
3	0,4621	2,1307	0,2936	0,390E-4
4	0,4621	2,1300	0,2935	0,298E-4

da reinicialização do método, na tabela 4.19 apresentamos os resultados com 5 iterações, variando o número de reinicializações . Para $W_i = 0$, com uma reinicialização (IRREST=2), ocorreu uma significativa redução no custo e um grande aumento em A, tendendo para os valores analíticos, embora z tenha dobrado seu valor. Os resultados convergem à partir da segunda reinicialização, que em tempo de computação corresponde aproximadamente a 15 iterações. Ainda com $W_i = 0$, para IRREST = 4, o valor da sobre-elevação é 0,40 e o tempo de resposta é 0,1 seg. Com $W_i = 1000$ o valor de A não se altera com as reinicializações do método, porém há uma pequena redução em z e no custo, na terceira reinicialização. Portanto neste exemplo o uso de reinicializações é significativo apenas quando $W_i = 0$, pois a melhoria obtida quando $W_i = 1000$ é muito pequena, em função do aumento resultante no tempo de computação.

4.6 - DETERMINAÇÃO DE PARÂMETROS EM UM PROCESSO QUÍMICO.

Com o objetivo de ilustrar outra aplicação do algorítmo desenvolvido neste trabalho, usaremos o GRAMIN para a determinação das constantes de reação na desidrogenação pirolítica do benzeno em difenil e trifenil, a partir de dados experimentais sobre a cinética do processo. Este problema foi resolvido por Lápidus(57) , com o método de quasilinearização (6,57). Empregaremos estes dados na comparação com os resultados do GRAMIN.

A reação química que descreve o processo é:

$$2\ C_6H_6 \rightleftharpoons C_{12}H_{10} + H_2$$

$$C_6H_6 + C_{12}H_{10} \rightleftharpoons C_{10}H_{14} + H_2$$

Para esta reação, Lãpidus utilizou o seguinte modelo cinético para um reator integral de fluxo:

$$\dot{x}_1 = -r_1 - r_2 \quad ; \; x_1(0) = 1 \qquad (4.6.1)$$

$$\dot{x}_2 = \frac{r_1}{2} - r_2 \quad ; \; x_2(0) = 0 \qquad (2.6.2)$$

onde:

$$r_1 = k_1\left[x_1^2 - x_2(2 - 2x_1 - x_2)/3K_1\right] \qquad (4.6.3)$$

$$r_2 = k_2\left[x_1x_2 - (1 - x_1 - 2x_2)(2 - 2x_1 - x_2)/9K_2\right] \qquad (4.6.4)$$

e

$$x_1 = \left(\frac{\text{libra-mol de benzeno}}{\text{libra-mol de benzeno puro de alimentação}}\right)$$

$$x_2 = \left(\frac{\text{libra-mol de difenil}}{\text{libra-mol de benzeno puro de alimentação}}\right)$$

k_1, k_2 = constantes de reação que se deseja estimar;

K_1, K_2 = constantes de equilíbrio .

A variável independente t é o tempo espacial ou o recíproco da velocidade espacial. É o tempo necessário para processar um volume de alimentação igual ao volume do reator, medido em condições determinadas (58). Esta variável tem a dimensão de tempo.

Os dados experimentais sobre a cinética do processo estão mostrados abaixo na tabela 4.20.

TABELA 4.20 - VALORES EXPERIMENTAIS DE x_1 E x_2 (referência 57)

t	5,63E-4	11,32E-4	16,97E-4	22,62E-4	34,00E-4	39,70E-4	45,20E-4	169,7E-4
Y_1	0,828	0,704	0,622	0,565	0,499	0,482	0,470	0,443
Y_2	0,0737	0,113	0,1322	0,1400	0,1468	0,1477	0,1477	0,1477

Estes dados são valores médios obtidos após uma série de medidas experimentais. As variáveis y_1 e y_2 estão relacionadas com as variáveis de estado x_1 e x_2 através das seguintes expressões:

$$y_1 = x_1 + \text{erro} \tag{4.6.5}$$

$$y_2 = x_2 + \text{erro} \tag{4.6.6}$$

O problema a ser resolvido é a determinação de k_1 e k_2, de forma a minimizar este erro.

No algorítmo de quasilinearização Lâpidus utilizou o seguinte critério de custo:

$$S = \sum_{r=1}^{8} \{ [y_1(t_r) - x_1(t_r;\underline{u})]^2 + [y_2(t_r) - x_2(t_r;\underline{u})]^2 \} \tag{4.6.7}$$

com $\underline{u} = [u_1, u_2]^T = [k_1, k_2]^T$; onde t_r ($r = 1,2,\ldots,8$) representa os valores do tempo espacial nos quais as medidas foram tomadas.

O critério de custo definido pela equação (4.6.7) é computado apenas nos valores de t, para o quais se efetuaram as medidas. Entretanto para a plicação do GRAMIN, o custo deve estar na forma da equação (2.4.7):

$$J(\underline{u}) = \phi(\underline{x}(t_f,\underline{u})) \ .$$

Assim sendo, vamos usar um funcional de custo da forma:

$$J(\underline{u}) = \int_{t_0}^{t_f} P(t) \{ [y_1(t) - x_1(t,\underline{u})]^2 + [y_2(t) - x_2(t,\underline{u})]^2 \} \, dt \tag{4.6.8}$$

onde $P(t) = 0$ para $t \neq t_r$

$P(t) = 1$ para $t = t_r$

e $[t_0, t_f]$ é o intervalo de tempo espacial de operação do sistema.

Definindo uma nova variável de estado:

$$\dot{x}_3 = P(t)\{[y_1(t) - x_1(t,\underline{u})]^2 + [y_2(t) - x_2(t,\underline{u})]^2\} \quad (4.6.9)$$

e $x_3(0) = 0$

a expressão para o funcional de custo torna-se:

$$J(\underline{u}) = \phi(\underline{x}(t_f,\underline{u})) = x_3(t_f,\underline{u}) \quad (4.6.10)$$

Portanto o estado do sistema fica definido pelas equações (4.6.1), (4.6.2) e (4.6.9). Para estas equações de estado a equação adjunta é:

$$\dot{\lambda}_1 = (-2u_1z_1 + u_2z_2)\lambda_1 + (2u_1z_3 + u_2z_4)\lambda_2 \quad (4.6.11)$$

$$\dot{\lambda}_2 = (-u_1z_1 + u_2z_2)\lambda_1 + (-u_1z_3 + u_2z_4)\lambda_2 \quad (4.6.12)$$

$$\dot{\lambda}_3 = 2P(t)(y_1 - x_1)\lambda_1 + 2P(t)(y_2 - x_2)\lambda_2 \quad (4.6.13)$$

com $\underline{\lambda}(t_f) = [0 \quad 0 \quad 1]^T \quad (4.6.14)$

onde $z_1 = x_1 + \dfrac{x_2}{3u_1} \quad (4.6.15)$

$$z_2 = \frac{4(1-x_1) - 5x_2}{9u_2} \quad (4.6.16)$$

$$z_3 = \frac{x_1 + x_2 - 1}{3u_1} \quad (4.6.17)$$

$$z_4 = x_1 + \frac{5(1-x_1) - 4x_2}{9u_2} \quad (4.6.18)$$

Finalmente a expressão para o gradiente é :

$$\underline{g}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \int_{t_0}^{t_f} \left\{ \frac{\partial f_1(t)}{\partial u_1} \lambda_1(t) + \frac{\partial f_2(t)}{\partial u_1} \lambda_2(t) + \frac{\partial f_3(t)}{\partial u_1} \lambda_3(t) \right\} dt \\ \int_{t_0}^{t_f} \left\{ \frac{\partial f_1(t)}{\partial u_2} \lambda_1(t) \quad \frac{\partial f_2(t)}{\partial u_2} \lambda_2(t) + \frac{\partial f_3(t)}{\partial u_2} \lambda_3(t) \right\} dt \end{bmatrix} \qquad (4.6.19)$$

onde: $$\frac{\partial f_1}{\partial u_1} = -\left\{ x_1^2 - x_2 \frac{(2 - 2x_1 - x_2)}{3u_1} \right\} \qquad (4.6.20)$$

$$\frac{\partial f_2}{\partial u_1} = -\frac{1}{2} \frac{\partial f_1}{\partial u_1} \qquad (4.6.21)$$

$$\frac{\partial f_3}{\partial u_1} = \frac{\partial f_3}{\partial u_2} = 0 \qquad (4.6.22)$$

$$\frac{\partial f_1}{\partial u_2} = \frac{\partial f_2}{\partial u_2} = -\left\{ x_1 x_2 - \frac{(1 - x_1 - 2x_2)(2 - 2x_1 - x_2)}{9u_2} \right\} \qquad (4.6.23)$$

O programa para este exemplo está listado no apêndice A7.

Na tabela 4.20, os valores de t indicam uma distribuição das medidas no intervalo de 0 a $200X10^{-4}$. Para abranger exatamente todos os valores de t desta tabela, seriam necessários 20 001 pontos no intervalo de discretização do GRAMIN. Esta discretização tornaria o tempo de computação cerca de 100 vezes maior, do que se fossem utilizados 201 ponto neste intervalo, que é o dimensionamento máximo adotado no GRAMIN, para matrizes e vetores. Em vista disto, construímos o gráfico da figura 4.13 à partir dos dados da tabela 4.20 e interpolamos os valores de y_1 e y_2, de forma a

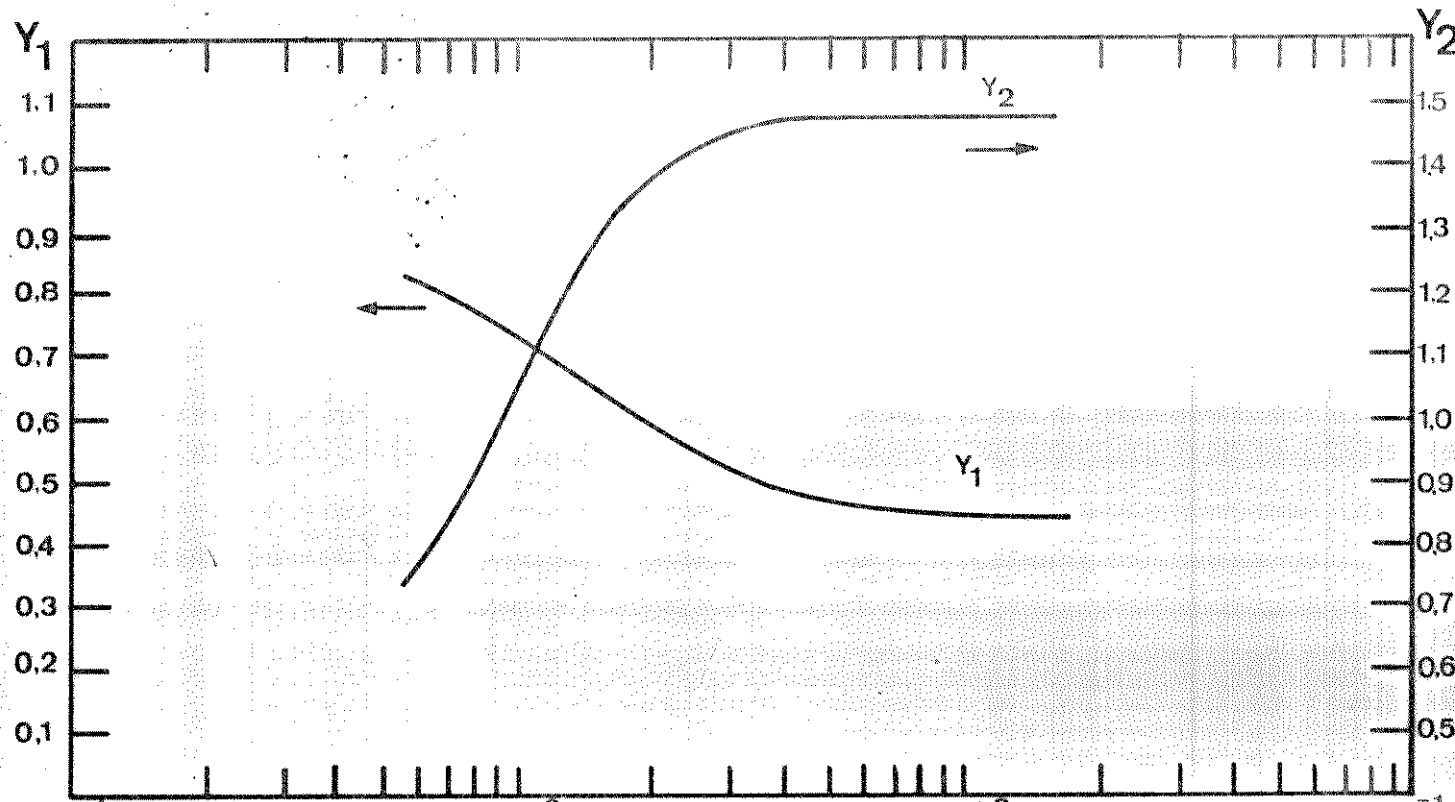


Fig. 4.13 - Curvas construídas à partir das medidas experimentais Y_1 e Y_2 das variáveis de estado X_1 e X_2 .

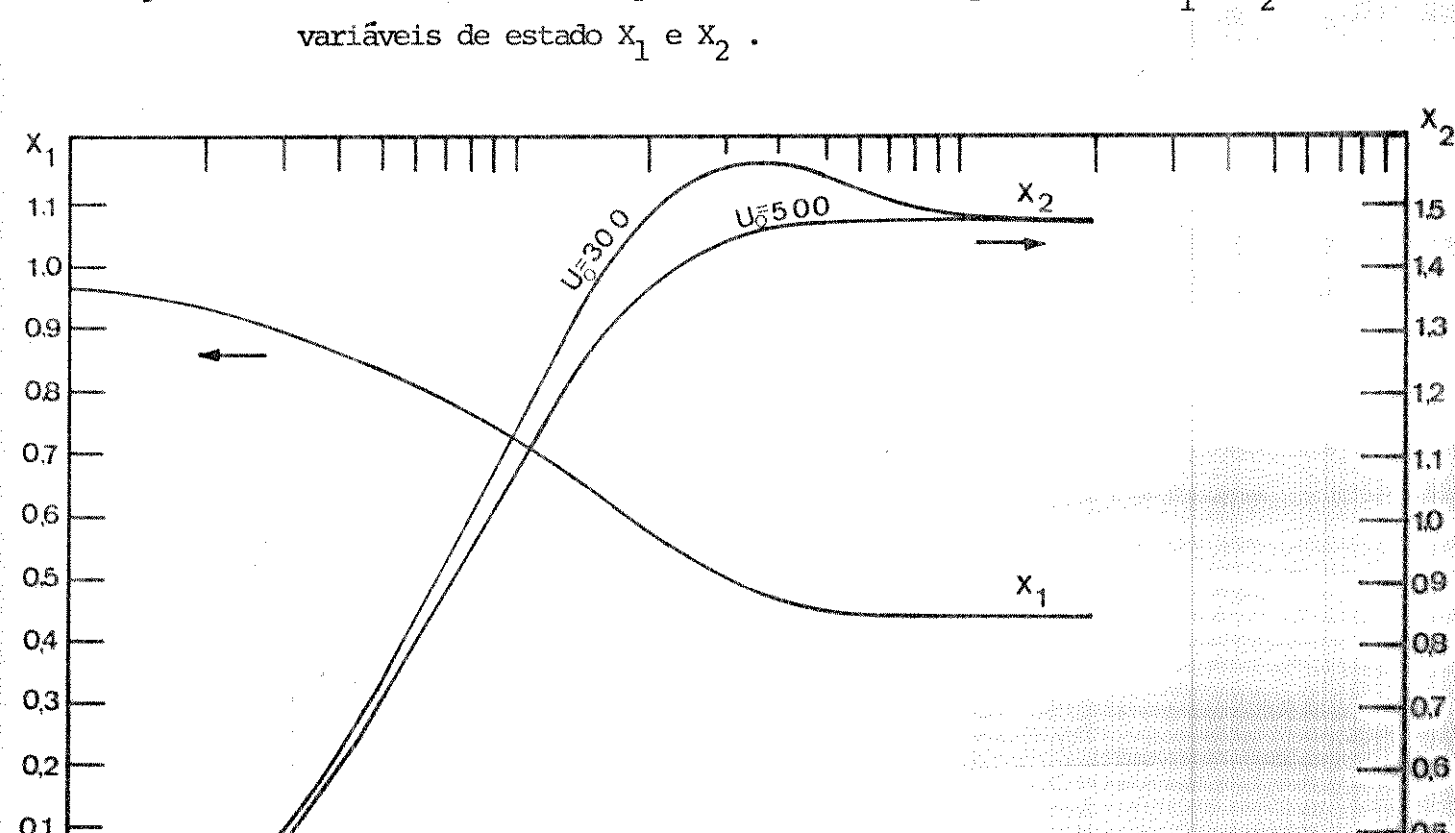


Fig. 4.14 - Curvas para as variáveis de estado X_1 e X_2 obtidas com o GRAMIN.

expressar t como múltiplo de 0,0001, permitindo a utilização de 201 pontos no intervalo de discretização. Os valores resultantes desta interpolação, que serão usados no funcional de custo, estão na tabela 4.21 abaixo.

TABELA 4.21 - VALORES DE Y_1 E Y_2 OBTIDOS POR INTERPOLAÇÃO

t	6E-4	11E-4	17E-4	23E-4	34E-4	40E-4	45E-4	170E-4
Y_1	0,815	0,711	0,622	0,559	0,499	0,482	0,471	0,443
Y_2	0,765	0,1105	0,1322	0,1405	0,1468	0,1477	0,1477	0,1476

Na execução do programa, usamos os mesmos valores adotados por Lãpidus (57) para as constantes de equilíbrio: $K_1 = 0,242$ e $K_2 = 0,428$.

O GRAMIN foi executado com diferentes estimativas iniciais u_0, para determinar a região de convergência do método. Os resultados estão mostrados na tabela 4.22 e foram obtidos com 4 iterações sem reinicialização. Na mesma tabela estão os resultados de Lãpidus, também com 4 iterações.

Nesta tabela observamos que o GRAMIN apresenta uma região de convergência maior que a do método de quasilinearização, inclusive convergindo à partir da estimativa inicial $\underline{u}_0 = 0$. Esta característica é útil, quando não se tem uma idéia da região em que o valor ótimo está localizado.

Outra característica observada é que o GRAMIN não apresenta divergência, como ocorre com o método de quasilinearização, em nenhum dos casos testados. Quando o método não converge, ele permanece estacionário no ponto inicial $\underline{u}_0$. Esta condição estacionária ocorre, porque o valor do funcional de custo resulta sempre em

TABELA 4.22 - EFEITO DA ESTIMATIVA INICIAL u_0 SOBRE A CONVERGÊNCIA.

NPTOS = 200 NITER = 4 IRREST = 1 $\Theta = 10^8 t$

valores iniciais		valores finais		GRAMIN
u_1^0	u_2^0	u_1	u_2	J(u)
0	0	407,636	57,235	225,587
100	100	390,341	183,542	57,454
300	300	375,019	318,798	7,349
500	500	368,387	438,311	3,304
1000	1000	**	**	**
valores de Lápidus (Quasilinearização)				
100	100	***	***	*
300	300	347,443	403,132	*
500	500	347,402	402,751	*
1000	1000	***	***	*

* DADOS NÃO DISPONÍVEIS.

** NÃO CONVERGIU.

*** DIVERGIU

zero, à despeito dos sucessivos passos na busca unidimensional.Já haviamos deparado com este problema, quando de início tentamos reproduzir os valores de Lápidus, usando o intervalo de tempo espacial de 0 a $200X10^{-4}$ e obtivemos custo zero para qualquer valor inicial. Isto ocorria porque, sendo o tempo medido em horas, o intervalo de integração escolhido era muito pequeno e causava problemas no Runge-Kutta. Uma mudança na escala de tempo do tipo: $\theta = at$ resolveu o problema. Com $a=10^4$, quatro iterações, 201 pontos no intervalo de discretização e $\underline{u}_0=300$, obtivemos os mesmos valores finais de $\underline{u}$ da tabela 4.22, porém com um custo $J(\underline{u}) = 0,0057$. Os valores da tabela 4.22 foram obtidos com $a=10^8$, de onde se conclui que aumentando o fator de escala, o valor do funcional de custo aumenta, melhorando a sensitividade do método. Podemos supor que aumentando ainda mais o fator de escala, o método poderia convergir, mesmo com $u_0=1000$.

Na figura 4.14 estão as curvas para as variáveis de estado x_1 e x_2 em função do tempo espacial. Estas curvas foram construídas com $u_0=300$ e $u_0=500$. As curvas para x_1, com ambas as condições iniciais, são coicidentes. Ao passo que a curva x_2, com $u_0=500$, reproduz os valores experimentais da figura 4.13.

C A P I T U L O 5

COMENTÁRIOS FINAIS

O algorítmo implementado neste trabalho permite de uma maneira simples e direta, a obtenção de entradas de controle ótimo, sem considerar a linearidade do sistema e do funcional de custo . Desta forma é possível tratar sistemas lineares e não lineares indiferentemente. Sendo uma extensão do método do gradiente conjugado, o algorítmo mantêm suas características de estabilidade e convergência rápida. Se o controle está sendo calculado fora de linha (off-line), ou seja, os resultados são armazenados em um meio externo para uso posterior, tais como formulários, fita magnética ou algo semelhante, então o tempo de computação não é um fator decisivo. Neste caso é possível jogar com outros parâmetros para melhorar a qualidade do controle, tais como: número de iterações , reinicializações, intervalo de discretização e busca unidimensional. Dos exemplos do capítulo anterior, pode-se observar que é muito difícil determinar a combinação ótima destes parâmetros, para aplicação direta em qualquer problema de controle, porque a qualidade aceitável do controle varia de processo a processo, e também a topologia do funcional de custo.

Todos os casos discutidos no capítulo 4 estavam voltados para o controle fora de linha. Os casos de estimação de parâmetros, excetuando-se as aplicações em controle adaptativo, são inerentemente de natureza fora de linha. Entretanto o controle do motor DC se tornaria mais interessante, se pudesse ser aplicado em linha

(on-line) ao processo.O uso do algorítmo no controle digital direto de processos viria extender sobremaneira seu campo de aplicação.

É economicamente inviável, dedicar ao controle digital direto de processos uma máquina de grande porte, que custa de 10 a 100 vezes mais que o processo a ser controlado. Uma alternativa promissora é o emprego de microcomputadores. Para testar a viabilidade desta aplicação, o GRAMIN foi codificado no Basic interpretativo do Apple II (59) e no NE-Z8000 (60). O Apple II é um microcomputador com um interpretador Basic e sistema operacional em 12K de ROM (Read Only Memory ou Memória Apenas de Leitura(61)) , com 48K de RAM (Randon-Access Memory ou Memória de Acesso Direto(61)) disponíveis para o usuário. A CPU do microcomputador Apple é baseada no microprocessador 6502, que possui uma palavra de 8 bits. Esta capacidade de memória permite a execução do GRAMIN com seu dimensionamento máximo: sistemas definidos por 10 variáveis de estado e com 201 pontos no intervalo de discretização.O NE-Z8000 é uma máquina gêmea do Sinclair-ZX81, com um sistema operacional e interpretador Basic contidos em 8K de ROM e 16K de RAM disponíveis. A CPU do NE-Z8000 também possui palavra de 8 bits e é baseado no microprocessador Z80. Nesta máquina com capacidade limitada de memória, o GRAMIN pode ser usado em sistemas com um máximo de 4 variáveis de estado, e 44 pontos no intervalo de discretização.

A aplicação do algorítmo para controle digital direto, requer que as entradas de controle sejam calculadas, em uma fração do intervalo $[t_0,t_f]$ de operação do sistema. A fim de investigar esta possibilidade, na tabela 6.1 apresentamos o tempo de computação requerido pelas diferentes máquinas, para o controle do motor DC no espaço de funções contínuas. O programa foi executado sem reinicialização, com 10 iterações, 21 pontos para o intervalo de discretização, $W_i = 10\,000$ e $u_0(t) = 0$.

TABELA 6.1 - TEMPO DE CPU PARA O CASO DE FUNÇÕES CONTÍNUAS.

computador	tempo de compilação seg	tempo de processamento seg
DEC SYSTEM 10	22	10
IBM 370/145	300 † 840 ††	128 † 200 ††
HP 2001	420	372
APPLE	*	13006
NE-Z8000	*	17800

† - Batch

†† - Terminal

* - Linguagem interpretada

NITER = 10 NPTOS = 20 IRREST = 0 W_i = 10000 $u_0(t) = 0$

Da tabela 6.1 pode-se observar que os tempos de computação, requeridos pelos microcomputadores, inviabilizam seu uso com o GRAMIN para o controle digital direto. Parte deste tempo excessivamente longo é devido ao uso do Basic Interpretado. Este tempo poderia ser reduzido de 12 a 15 vezes com o uso de Forth (62) , uma linguagem compilada especial para a aplicação de microcomputadores ao controle de processo. Estimativamente, empregando o compilador Forth e fazendo apenas 5 iterações do método, o tempo de computação no Apple II cai de 13000 segundos para 430 seg. Portanto a aplicação do GRAMIN ao controle digital direto de processos,em vista da velocidade atual dos microcomputadores, só é viável se o intervalo de tempo de operação do sistema for superior a 20 000 segundos. Além disso este sistema deve apresentar uma dinâmica tal, que o controle não necessite ser atualizado, em tempos menores que 1/20 deste intervalo.

Uma alternativa ao controle digital direto, e que gostaríamos de deixar como sujestão para um posterior trabalho, é o controle híbrido direto. Neste caso teríamos uma máquina híbrida, onde a integração das equações de estado e adjunta, seria realizada analógicamente, sob controle da parte digital, que também se encarregaria dos demais processamentos. Isto representaria uma economia no tempo de computação, considerando que a maior parte do tempo despendido no GRAMIN é gasto realizando as integrações através do algorítmo de Runge-Kutta. Cremos ser possível com este arranjo, o controle ótimo em tempo real de processos, com dinâmica semelhante ao motor DC do capítulo anterior.

RELAÇÃO DOS APÊNDICES

A P Ê N D I C E A1

BUSCA UNIDIMENSIONAL

Em uma das etapas do método do Gradiente Conjugado, e a cada iteração deste, deve ser encontrado o escalar $\alpha_i > 0$ que minimiza $F(x_i+\alpha_i p_i)$ equação (2.3.17), que é equivalente a encontrar x_i^*, tal que $x_i^* = x_i+\alpha_i p_i$, onde x_i^* é o argumento ótimo ou seja, produz o valor mínimo do funcional na iteração i, e α_i é o passo ótimo da busca. A fim de simplificar a notação não iremos carregar o índice i, nos algorítmos apresentados a seguir.

Os métodos numéricos que procuram o α_i ótimo, são chamados de BUSCA UNIDIMENSIONAL (28,36,37). Entre os tipos de Busca, encontram-se Buscas Sequenciais e não Sequenciais. Segundo Himmenlblau (37) as buscas do tipo Sequencial são mais eficientes, e as melhores dentre estas são as de Fibonacci e a de Secção Aurea. A título de comparação, a Busca Uniforme, que é do tipo não Sequencial, requer 199 cálculos da função objetivo, para reduzir o intervalo por um fator de 5×10^{-3}, enquanto as de Fibonacci e a de Secção Aurea requerem 11 cálculos para reduzir o intervalo por aquele mesmo fator.

Outros tipos de buscas que apresentam uma eficiência ainda maior, realizam a minimização da função objetivo por meio de extrapolação e interpolação(37). Entre estes métodos tem-se o de Davies-Swann-Campey (DSC) e o de Powell. A seguir apresentamos os algorítmos de cada um destes métodos.

I - ALGORÍTMO DE DAVIES-SWANN-CAMPEY (DSC).

Na figura A1.1 está ilustrado o procedimento de busca através

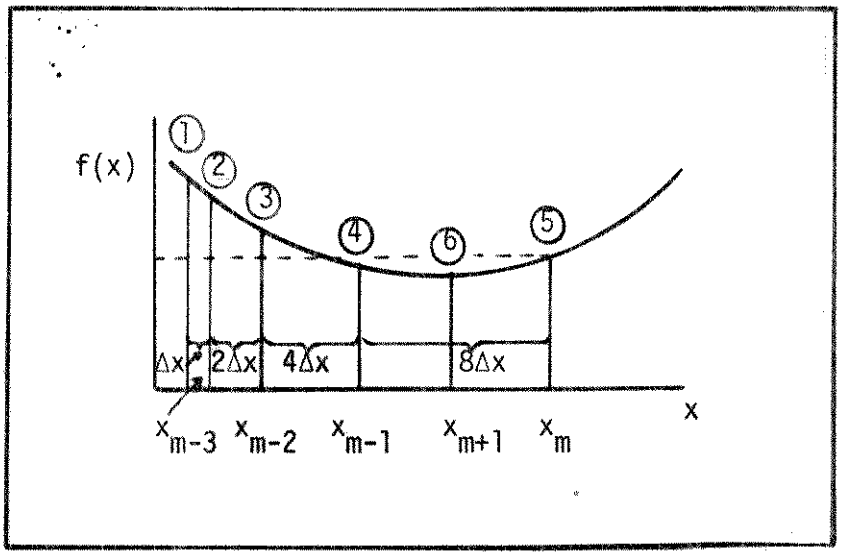


Fig. A1.1 - Algorítmo DSC para minimização.

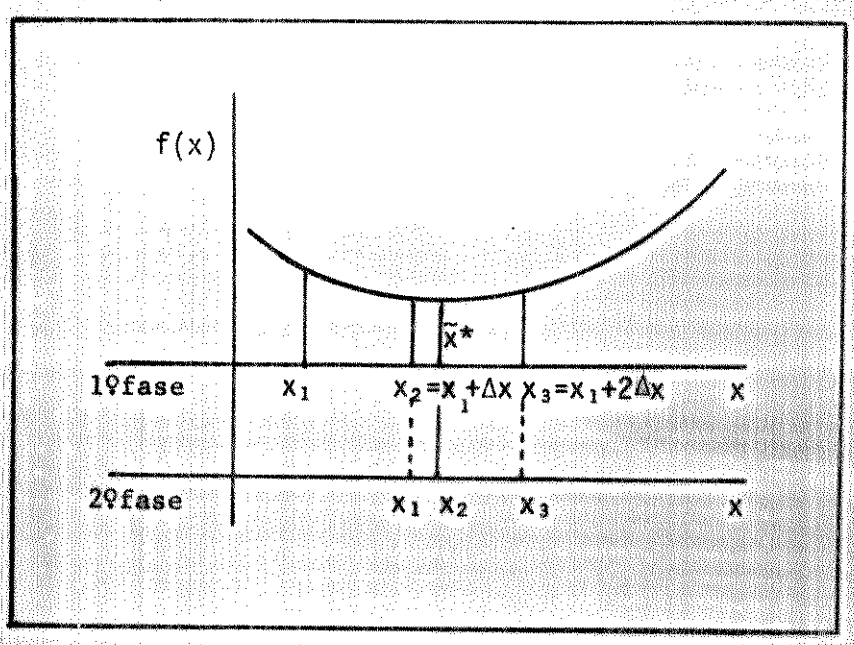


Fig. A1.2 - Algorítmo de Powell para minimização.

do algorítmo DSC. Nesta busca, partindo-se de um ponto x_0 inicial e de um passo inicial $\Delta x = \alpha p$, primeiramente determina-se o sentido de decrescimento da função, verificando se $F(x_0+\Delta x)$ ou se $F(x_0-\Delta x)$ é menor que $F(x_0)$. Uma vez identificado o sentido de decrescimento, escolhe-se $\Delta x = +\Delta x$ ou $-\Delta x$, de forma a caminhar na direção em que a função decresce. Em seguida os seguintes passos são efetuados:

1-Com k=0 tem-se $x_k=x_0$;

2-Fazendo $x_{k+1} = x_k+\Delta x$, calcula-se $F(x_{k+1})$;

3-Se $F(x_{k+1}) < F(x_k)$ dobra-se o passo Δx, e retorna-se ao passo 2 com k = k + 1. Se $F(x_{k+1}) > F(x_k)$ toma-se $x_m = x_{k+1}$, $x_{m-1} = x_k$, $x_{m-2} = x_{k-1}$, $x_{m+1} = x_k+\Delta x/2$ e calcula-se $F(x_{m+1})$.

4-Escolhe-se 3 dos 4 argumentos x_{m+i} ,para i = -2,-1,0,1 que produzem os menores valores de $F(x_{m+i})$. Destes três elementos, toma-se x_b igual ao elemento central, $x_a = x_b-\Delta x/2$ e $x_c = x_b+\Delta x/2$.

5- Por estes pontos interpola-se uma função quadrática, para estimar o mínimo x*. Onde $x^* \simeq \tilde{x}^*$ e

$$\tilde{x}^* = x_b + \frac{\Delta x \left[F(x_a) - F(x_c) \right]}{4 \left[F(x_a) - 2F(x_b) + F(x_c) \right]} \qquad (A1\text{-}1)$$

6-Os pontos $\tilde{x}^*$ ou x_c podem ser escolhidos como aproximação do mínimo, dependendo de qual dos argumentos produza o menor valor da função objetivo. Se desejado, pode-se melhorar a solução, tomando o ponto assim encontrado como x_0, e voltando ao passo 1 reinicializar o método.

II - O ALGORÍTMO DE POWELL.

O método de Powell difere do método DSC de busca, pelo fato

que este último faz a interpolação quadrática somente após determinar o intervalo ("bracket") onde o mínimo está localizado, enquanto que o método de Powell faz sucessivas interpolações quadráticas a cada 3 pontos, independente de ter localizado o intervalo onde o mínimo está situado. Outra diferença é que no algorítmo DSC os pontos usados para interpolação estão igualmente espaçados, resultando em uma fórmula de interpolação quadrática mais simples. No método de Powell os pontos não são igualmente espaçados e a fórmula de interpolação quadrática é mais complexa.

A figura A1.2 ilustra o procedimento de busca pelo algorítmo de Powell, que é descrito a seguir.

1-Partindo-se de um ponto x_1, calcula-se $x_2=x_1+\Delta x$, $F(x_1)$e $F(x_2)$.

2-Se $F(x_1) > F(x_2)$, toma-se $x_3=x_1+2\Delta x$. Se $F(x_1) \leq F(x_2)$, toma-se $x_3=x_1-\Delta x$.

3-Calcula-se $F(x_3)$.

4-Estima-se o valor do mínimo $\tilde{x}^*$ por:

$$\tilde{x}^* = -\frac{1}{2}\,\frac{(x_2^2 - x_3^2)F(x_1) + (x_3^2 - x_1^2)F(x_2) + (x_1^2 - x_2^2)F(x_1)}{(x_2 - x_3)F(x_1) + (x_3 - x_1)F(x_2) + (x_1 - x_2)F(x_1)} \qquad \text{(A1.2)}$$

5-Termina-se a busca quando $\tilde{x}^*$ e qualquer um dos $\{x_1, x_2, x_3\}$ correspondente ao menor F(x), diferir de um certo valor preestabelecido.

O critério de parada poderia também ser em função de $F(\tilde{x}^*)$ e do menor dos F(x). Se a condição de parada do algorítmo não for satisfeita, $F(\tilde{x}^*)$ deve ser calculada e , discarta-se o elemento do conjunto $\{x_1, x_2, x_3\}$ correspondente ao maior valor de F(x), retornado-se em seguida ao passo 4 com $\tilde{x}^*$ e os dois elementos restantes. Pode ocorrer que ao se discartar de um elemento do conjunto $\{x_1, x_2, x_3\}$, o intervalo em que o mínimo está localizado seja perdido. Nes

te caso deve ser escolhido outro elemento para o descarte, de forma a manter tal intervalo.

III - OUTROS ALGORÍTMOS.

A combinação dos métodos DSC e de Powell, gera algorítmos mais eficientes que o uso de cada um separadamente. Uma combinação frequentemente empregada é a que consiste em se tomar inicialmente os procedimentos do algorítmo DSC, até encontrar o intervalo em que o mínimo está localizado, a partir deste ponto utiliza-se os passos 4 e 5 do algorítmo de Powell. Exemplos de algorítmos que usam este tipo de combinação são encontrados em: Athans e colaboradores (55), Hasdorff (28) e Fletcher & Reeves (44), este último usa uma interpolação cúbica em lugar da quadrática. Nestes algorítmos, se for feita uma boa estimativa do passo inicial Δx, não é necessário fazer mais que uma interpolação para se obter uma razoável aproximação de $\tilde{x}^*$. Este é um fato importante, porque desta forma consegue-se uma apreciável redução no tempo de computação.

A P Ê N D I C E A2

EQUAÇÃO ADJUNTA

Neste apêndice mostramos que $\lambda(t)$, definido na equação (2.4.29) é solução da equação adjunta (2.4.32).

Equação (2.4.29): $\lambda(t) = \Phi^{\dagger}(t_f,t)\nabla_x(\underline{x}(t_f,u))$ (A2-1)

Equação (2.4.32): $\dot{\lambda}(t) = -f_x^{\dagger}\,\lambda(t)$ (A2-2a)

$$\lambda(t_f) = \nabla_x\phi(\underline{x}(t_f,u)) \tag{A2-2b}$$

Na expressão (A2-1) $\Phi(t_f,t)$ é a matriz de transição (5) e satisfaz as equações (2.4.24) e (2.4.25), ou seja:

De (2.4.24): $$\frac{d\Phi(t,t_0)}{dt} = A(t)\Phi(t,t_0) \tag{A2-3}$$

com $A(t) = f_x$ (equação 2.4.19) temos:

$$\frac{d\Phi(t,t_0)}{dt} = f_x\Phi(t,t_0) \tag{A2-4}$$

e de (2.4.25): $\Phi(t_0,t_0) = I$ (I é matriz identidade).

Para atingir o objetivo deste apêndice, vamos primeiramente mostrar que $\Phi(t,\tau)$ satisfaz também a equação:

$$\frac{d\Phi(t,\tau)}{d\tau} = -\Phi(t,\tau)A(\tau) \text{ tal que } \Phi(t,t) = I$$

$$e\ \tau\ e\ [t_0,t_f]$$

De (2.4.22) temos a equação diferencial para sistemas lineares variantes no tempo:

$$\dot{\underline{x}} = A(t)\underline{x} + \underline{B}(t)u \ . \qquad \text{(A2-5)}$$

Considerando o sistema livre, segue que $u = 0$ para todo $t \geq t_0$, ou:

$$\dot{\underline{x}} = A(t)\underline{x}$$

usando (2.4.23), obtemos:

$$\underline{x}(t) = \Phi(t,t_0)\underline{x}(t_0) \qquad \text{(A2-6)}$$

$$\underline{x}(\tau) = \Phi(\tau,t_0)\underline{x}(t_0) \qquad \text{(A2-7)}$$

fazendo $t_0 = \tau$ em (A2-6)

$$x(t) = \Phi(t,\tau)\underline{x}(\tau) \ . \qquad \text{(A2-8)}$$

Substituindo (A2-7) em (A2-8):

$$\underline{x}(t) = \Phi(t,\tau)x(\tau) = \Phi(t,\tau)\Phi(\tau,t_0)\underline{x}(t_0) \ . \qquad \text{(A2-9)}$$

Comparando com (A2-6) e considerando que estas relações devem ser válidas para qualquer $\underline{x}(t_0)$ segue:

$$\Phi(t,t_0) = \Phi(t,\tau)\Phi(\tau,t_0) \qquad \text{(A2-10)}$$

derivando esta expressão em relação a τ, temos,

$$\frac{d\Phi(t,\tau)}{d\tau}\,\Phi(\tau,t_0) + \Phi(t,\tau)\frac{d\Phi(\tau,t_0)}{d\tau} = 0 \ . \qquad \text{(A2-11)}$$

Em (A2-3) fazendo a mudança de variável $t \to \tau$:

$$\frac{d\Phi(\tau,t_0)}{d\tau} = A(\tau)\Phi(\tau,t_0) \qquad \text{(A2-12)}$$

Substituindo em (A2-11),

$$\{\frac{d\Phi(t,\tau)}{d\tau} + \Phi(t,\tau)A(\tau)\}\ \Phi(\tau,t_0) = 0 \qquad \text{(A2-13)}$$

Esta equação é válida para todo τ , em particular para $\tau = t_0$ temos $\Phi(t_0,t_0) = I$, então segue que:

$$\frac{d\Phi(t,\tau)}{d\tau} + \Phi(t,\tau)A(\tau) = 0 \qquad \text{(A2-14)}$$

ou

$$\frac{d\Phi(t,\tau)}{d\tau} = -\Phi(t,\tau)A(\tau) \text{ e } \Phi(t,t) = I \ . \qquad \text{(A2-15)}$$

Em (A2-15) com $t = t_f$ e $\tau = t$, obtemos:

$$\frac{d\Phi(t_f,t)}{dt} = -\Phi(t_f,t)A(t) \text{ e } \Phi(t_f,t_f) = I \qquad \text{(A2-16)}$$

tomando a transposta

$$\frac{d\Phi^{\dagger}(t_f,t)}{dt} = -(\Phi(t_f,t)A(t))^{\dagger} = -A^{\dagger}(t)\Phi^{\dagger}(t_f,t) \qquad \text{(A2-17)}$$

$$\text{e } \Phi^{\dagger}(t_f,t_f) = I \ .$$

Voltando em (A2-1) e derivando com respeito a t segue:

$$\dot{\lambda}(t) = \frac{d\lambda(t)}{dt} = \frac{d\Phi^{\dagger}(t_f,t)}{dt} \nabla_x\phi(x(t_f,u)) + \Phi^{\dagger}(t_f,t) \frac{d\nabla_x\phi(x(t_f,u))}{dt} \qquad \text{(A2-18)}$$

Como $\nabla_x \phi(x(t_f,u))$ não é função de t, segue:

$$\dot{\lambda}(t) = \frac{d\Phi^{\dagger}(t_f,t)}{dt} \nabla_x \phi(x(t_f,u)) \quad . \qquad \text{(A2-19)}$$

usando o resultado obtido em (A2-17) temos:

$$\dot{\lambda}(t) = -A^{\dagger}(t)\underbrace{\Phi^{\dagger}(t_f,t)\nabla_x \phi(x(t_f,u))}_{\lambda(t)} \qquad \text{(A2-20)}$$

Comparando com (A2-1) e usando (2.4.19) temos portanto que $\lambda(t)$ satisfaz a equação (A2-2a), ou seja:

$$\boxed{\dot{\lambda}(t) = -f_x^{\dagger}\lambda(t)}$$

A condição terminal (A2-2b) é obtida de (A2-1) com $t = t_f$:

$$\lambda(t_f) = \Phi^{\dagger}(t_f,t_f)\nabla_x \phi(x(t_f,u)) \qquad \text{e} \qquad \Phi(t_f,t_f) = I \qquad \text{(A2-21)}$$

Logo:

$$\boxed{\lambda(t_f) = \nabla_x \phi(x(t_f,u))}$$

APÊNDICE A3: PROGRAMA PRINCIPAL E SUB-ROTINAS AUXILIARES PARA O CONTROLE DO MOTOR DC NO ESPAÇO DE FUNÇÕES CONTÍNUAS.

```
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
C     INICIALIZACAO
C     DO 115 L3=1,4
      MODE=1
      MODO=1
      N=4
      M=1
        TYPE 333
333     FORMAT(' ENTRE COM WI')
        ACCEPT 334,WI
334     FORMAT(G)
        WRITE(3,335)WI
335     FORMAT(' WI= ',G)
      R=5.
      W(1)=WI
      W(2)=WI
      W(3)=WI
      TI=0.
      TF=5.
        TYPE 336
336     FORMAT(' ENTRE COM NITER ')
        ACCEPT 337,NITER
337     FORMAT(G)
        WRITE (3,338)NITER
338     FORMAT(' NITER= ',G)
        TYPE 339
339     FORMAT(' ENTRE COM NPTOS ')
        ACCEPT 337,NPTOS
        WRITE(3,340)NPTOS
340     FORMAT(' NPTOS= ',G)
      NPTOSG=NPTOS
C
C     FIM DA INICIALIZACAO
      DO 10 I=1,N
10    X0(I)=0.
        TYPE 341
341     FORMAT( ' ENTRE COM U0 ')
        ACCEPT 337, UI
        WRITE(3,342),UI
342     FORMAT(' U0= ',G)
      DO 20 I=1,201
20    U(I)=UI
      NPT=NPTOS+1
        TYPE 345
345     FORMAT(' ENTRE COM IRREST ')
        ACCEPT 337,IRT
        WRITE(3,346)IRT
346     FORMAT(' IRREST= ',G)
      DO 41 IRREST=1,IRT
      CALL GRAMI
      DO 30 I=1,NPT
30    WRITE(3,100) I,X(1,I),X(2,I),X(3,I),X(4,I),U(I)
```

```
      WRITE(3,344)IRREST
41    CONTINUE
100   FORMAT(2X,6G20.8)
344   FORMAT(' NUMERO DO RESTART= ',G14.8)
      WRITE(3,354)NITER,NPTOS,W(1),IRT
354   FORMAT(2X, ' NITER= ',I5,' NPTOS= ',I5,' WI= ',G14.8,' IREST
     1= ',I5)
115   CONTINUE
      END
C
      SUBROUTINE FUNX(T,B,F,UDT)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
      DIMENSION UDT(201),F(11)
      I=(T-TI)/DT+1
      F(1)=X1(2)
      F(2)=-X1(2)/3.+5.*X1(3)
      F(3)=-X1(3)+.1*UDT(I)
      F(4)=(X1(1)-10)**2+R*UDT(I)**2
      RETURN
      END
C
      SUBROUTINE FUNL(T,B,F,UDT)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
      DIMENSION UDT(201),F(11)
      I=(T-TI)/DT+1
      F(1)=-2*(X(1,I)-10)-X1(4)
      F(2)=-1*(X1(1)-1./3.*X1(2))
      F(3)=-1*(5*X1(2)-X1(3))
      F(4)=0
      RETURN
      END
C
      SUBROUTINE GRAXF
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
      X1(1)=2.*W(1)*(X1(1)-10.)
      X1(2)=2.*W(2)*X1(2)
      X1(3)=2.*W(3)*X1(3)
```

```
      X1(4)=1.
      RETURN
      END
C
      SUBROUTINE GRADI(GA,I,J)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMONS DO USUARIO
C
      COMMON R,W(3)
C
      GA=.1*ALBDA(3,I)+2*R*U(I)*ALBDA(4,I)
      RETURN
      END
      SUBROUTINE FI(AC)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUSARIO
C
       COMMON R,W(3)
C
      AC=X1(4)+W(1)*(X1(1)-10.)**2+W(2)*X1(2)**2+W(3)*X1(3)**2
      RETURN
      END
C
```

APÊNDICE A4:PROGRAMA PRINCIPAL E SUB-ROTINAS AUXILIARES PARA O CONTROLE DO MOTOR DC NO ESPAÇO DE ENTRADAS AMOSTRADAS.

```
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),S11(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
C     INICIALIZACAO
      MODE=0
C     PRODUTO ESCALAR DE VETORES
      MODO=3
C     ESPAÇO DE ENTRADAS AMOSTRADAS
      N=4
      M=8
      R=.5
      W(1)=1000
      W(2)=1000
      W(3)=1000
      TI=0.
      TF=8.
      NPTOS=160
      NPTOSG=8
      ID=3
      NITER=12
      NPT=NPTOS+1
      DO 10 I=1,N
10    X0(I)=0.
      DO 20 I=1,M
20    U(I)=0.
      DO 41 IRREST=1,1
      CALL GRAMI
      WRITE(3,101) U(1),U(2),U(3),U(4)
      WRITE(3,101)U(5),U(6),U(7),U(8)
101   FORMAT(2X,4G20.8)
      DO 30 I=1,NPT
30    WRITE(3,100) I,X(1,I),X(2,I),X(3,I),X(4,I),U(I)
41    CONTINUE
100   FORMAT(2X,6G20.8)
      END
C
      SUBROUTINE FUNX(T,N,F,UDT)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),S11(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
      DIMENSION UDT(201),F(11)
      IA=NPTOS/M
      I1=(T-TI)/DT
      K=I1/IA+1
      F(1)=X1(2)
      F(2)=-X1(2)/3.+5.*X1(3)
      F(3)=-X1(3)+.1*UDT(K)
      F(4)=(X1(1)-10)**2+R*UDT(K)**2
      RETURN
```

```
      END
C
      SUBROUTINE FUNCT(R,F,UDT)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
      DIMENSION UDT(201),F(11)
      I=(T-TI)/DT+1
      F(1)=-2*(X(1,I)-10)*X1(4)
      F(2)=-1*(X1(1)-1./3.*X1(2))
      F(3)=-1*(5*X1(2)-X1(3))
      F(4)=0
      RETURN
      END
C
      SUBROUTINE GRAXF
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
      X1(1)=2.*W(1)*(X1(1)-10.)
      X1(2)=2.*W(2)*X1(2)
      X1(3)=2.*W(3)*X1(3)
      X1(4)=1.
      RETURN
      END
C
      SUBROUTINE GRADI(GA,I,J)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
      IA=NPTOS/M
      I1=(I-TI)/DT
      K=I1/IA+1
      GA=.1*ALBDA(3,I)+2*R*U(K)*ALBDA(4,I)
      RETURN
      END
      SUBROUTINE FI(AC)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
```

```
C       COMMONS DO USUARIO
C
        COMMON R,W(3)
C
        AC=X1(4)+W(1)*(X1(1)-10.)**2+W(2)*X1(2)**2+W(3)*X1(3)**2
        RETURN
        END
```

APÊNDICE A5: PROGRAMA PRINCIPAL E SUB-ROTINAS AUXILIARES PARA O CONTROLE DO MOTOR DC COM ENTRADAS DE CONTROLE NO ESPAÇO DE MODULAÇÃO POR LARGURA DE PULSO.

```
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CI1(201),SI1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3),AKH
C
C
C     CASO DE M.L.P.
C     INICIALIZACAO
        TYPE 223
223     FORMAT( ' ENTRE COM ID')
        ACCEPT 224,ID
224     FORMAT(G)
        WRITE(ID,225)ID
225     FORMAT( ' ID= ',G)
      MODE=0
      MODO=4
      M=10
      TF=3.
        TYPE 565
565     FORMAT( ' ENTRE COM A')
        ACCEPT 566,AKH
566     FORMAT(G)
        TYPE 567,AKH
567     FORMAT(' A= ',G14.8)
        TYPE 333
333     FORMAT(' ENTRE COM WI,IPGK')
        ACCEPT 334,WI,IPGK
334     FORMAT(2G)
        WRITE(ID,335)WI,IPGK
335     FORMAT( ' WI,IPGK= ',2G )
      N=4
      R=5./(AKH**2)
      TI=0.
        W(1)=WI
        W(2)=WI
        W(3)=WI
        TYPE 339
339     FORMAT('ENTRE COM NPTOS ')
        ACCEPT 337,NPTOS
        WRITE(ID,340)NPTOS
340     FORMAT(' NPTOS= ',G)
      NPTOSG=10
      DO 10 I=1,N
10    X0(I)=0.
        TYPE 341
341     FORMAT(' ENTRE COM U0 ')
        ACCEPT 337,UI
        WRITE(ID,342),UI
342     FORMAT(' U0= ',G)
      NPT=NPTOS+1
        TYPE 336
336     FORMAT( ' ENTRE COM NITER ')
        ACCEPT 337,NITER
337     FORMAT(G)
        WRITE(ID,338)NITER
338     FORMAT('  NITER= ',G)
```

```
      TYPE 345
345   FORMAT(' ENTER COM IRREST')
      ACCEPT 337,IRT
      WRITE(ID,346)IRT
346   FORMAT(' IRREST= ',G)
      DO 41 IRREST=1,IRT
      CALL GRAMI
      WRITE(ID,101)U(1),U(2),U(3),U(4),U(5)
      WRITE(ID,101)U(6),U(7),U(8),U(9),U(10)
101   FORMAT(2X,5G20.8)
      DO 30 I=1,NPT
30    WRITE(ID,100)I,X(1,I),X(2,I),X(3,I),X(4,I)
      WRITE(ID,344)IRREST
41    CONTINUE
100   FORMAT(2X,5G20.8)
344   FORMAT( ' NUMERO DO RESTART= ',G14.8)
      WRITE(ID,354)NITER,NPTOS,W(1),IRT
354   FORMAT(2X,' NITER= ',I5,' NPTOS= ',I5,' WI= ',G14.8,'
     1 IREST= ',I5)
      END
C
      SUBROUTINE FUNX(T,N,F,UDT)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3),AKH
      DIMENSION UDT(201),F(11)
      I=(T-TI)/DT+1
      F(1)=X1(2)
      F(2)=-X1(2)/3.+5.*X1(3)
      F(3)=-X1(3)+.1*AKH*UDT(I)
      F(4)=(X1(1)-10)**2+R*(UDT(I)*AKH)**2
      RETURN
      END
C
      SUBROUTINE FUNL(T,N,F,UDT)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3),AKH
C
      DIMENSION UDT(201),F(11)
      I=(T-TI)/DT+1
      F(1)=-2*(X(1,I)-10)-X1(4)
      F(2)=-1*(X1(1)-1./3.*X1(2))
      F(3)=-1*(5*X1(2)-X1(3))
      F(4)=0
      RETURN
      END
C
      SUBROUTINE GRAXF
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
```

```
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
CC    COMMONS DO USUARIO
C
      COMMON R,W(3),AKH
C
      X1(1)=2.*W(1)*(X1(1)-10.)
      X1(2)=2.*W(2)*X1(2)
      X1(3)=2.*W(3)*X1(3)
      X1(4)=1.
      RETURN
      END
C
      SUBROUTINE GRAD1(GA,I,J)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
CC    COMMONS DO USUARIO
C
      COMMON R,W(3),AKH
C
        GA=.1*ALBDA(3,I)*AKH+ALBDA(4,I)*R*(AKH)**2*J
      RETURN
      END
      SUBROUTINE FI(AC)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
CC    COMMONS DO USUARIO
C
      COMMON R,W(3),AKH
C
      AC=X1(4)+W(1)*(X1(1)-10.)**2+W(2)*X1(2)**2+W(3)*X1(3)**2
      RETURN
      END
```

APÊNDICE A6: PROGRAMA PRINCIPAL E SUB-ROTINAS AUXILIARES PARA A DETERMINAÇÃO DE PARÂMETROS DE UM COMPENSADOR PÓLO ZERO.

```
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
C     INICIALIZACAO
        ID=3
      MODE=0.
C     PRODUTO ESCALAR NO RN
      MODO=2
C     ESPACO DE PARAMETROS
      TI=0.
      TF=2.
      W(1)=0.
      W(2)=0.
      W(3)=0.
      NPTOS=200
      NPTOSG=1
      N=4
      M=2
      NITER=0
      DO 10 I=1,8
10    X0(I)=0.
      U(1)=.365
      U(2)=2.23
        DT=2./NPTOS
      NPI=NPTOS+1
C     FIM DA INICIALIZACAO
      DO 41 IRREST=1,1
      CALL GRAMI
      WRITE(3,101)U(1),U(2)
101   FORMAT('U1= ',G20.8,'U2= ',G20.8)
      DO 30 I=1,NPI
30    WRITE(3,100)I,X(1,I),X(2,I),X(3,I),X(4,I)
41    CONTINUE
100   FORMAT(2X,5G20.8)
      END
C
C     FUNX
      SUBROUTINE FUNX(T,H,F,UDT)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
      DIMENSION UDT(201),F(11)
      F(1)=X1(2)
      F(2)=UDT(1)*34.2*(1.+X1(3)-X1(1))-2.23*X1(2)
      F(3)=(UDT(2)-5.)*(1.-X1(1))-5.*X1(3)
      F(4)=(1.-X1(1))**2
      RETURN
      END
```

```
C       FUNC
        SUBROUTINE FUNC(T,R,F,UDT)
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),S11(201),IPGK,ALP,ID
        COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        COMMON A(4),B(4),C(4),N11
C
C       COMMONS DO USUARIO
C
        COMMON R,W(3)
C
        DIMENSION UDT(201),F(11)
        I=(T-TI)/DT+1
        F(1)=34.2*UDT(1)*X1(2)+(UDT(2)-5.)*X1(3)+2.*(1.-X(1,I))*X1(4)
        F(2)=-X1(1)+2.23*X1(2)
        F(3)=-34.2*UDT(1)*X1(2)+5.*X1(3)
        F(4)=0.
        RETURN
        END
C
C       GRAXF
        SUBROUTINE GRAXF
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),S11(201),IPGK,ALP,ID
        COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        COMMON A(4),B(4),C(4),N11
C
C       COMMONS DO USUARIO
C
        COMMON R,W(3)
C
        X1(1)=2.*W(1)*(X1(1)-1.)
        X1(2)=2.*W(2)*X1(2)
        X1(3)=0.
        X1(4)=1.
        RETURN
        END
C
C       GRAD1
        SUBROUTINE GRAD1(GA,I,J)
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),S11(201),IPGK,ALP,ID
        COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        COMMON A(4),B(4),C(4),N11
C
C       COMMONS DO USUARIO
C
        COMMON R,W(3)
C
        IF(1-J)11,10,10
10      GA= 34.2*(X(3,I)-X(1,I)+1.)*ALBDA(2,I)
        GO TO 30
11      GA=(1.-X(1,I))*ALBDA(3,I)
30      CONTINUE
        RETURN
        END
C       FI
        SUBROUTINE FI(AC)
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),S11(201),IPGK,ALP,ID
```

```
      COMMON X0(10),T1,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(1),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
      COMMON R,W(3)
C
      AC=X1(4)+W(1)*(1.-X1(1))**2+W(2)*X1(2)**2
      RETURN
      END
```

APÊNDICE A7: PROGRAMA PRINCIPAL E SUB-ROTINAS AUXILIARES PARA A DETERMINAÇÃO DE PARÂMETROS CINÉTICOS EM UM PROCESSO QUÍMICO.

```
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
C
C     COMMONS DO USUARIO
C
        COMMON P(201),Y(2,201)  ,CK1,CK2
C
C     INICIALIZACAO
        ID=31
      MODE=0.
C     PRODUTO ESCALAR NO RN
      MODO=2
C     ESPACO DE PARAMETROS
      TI=0.
      TF=2000000.
      NPTOS=200
      NPTOSG=1
      N=3
      M=2
      NITER=4
        X0(1)=1
        X0(2)=0.
        X0(3)=0.
        U(1)=.00001
        U(2)=.00001
        DT=(TF-TI)/200
      NP1=NPTOS+1
        DO 40 I=1,NP1
        P(I)=0
        Y(1,I)=0
40      Y(2,I)=0
        P(6)=1
        Y(1,6)=.815
        Y(2,6)=.0765
        P(11)=1
        Y(1,11)=.711
        Y(2,11)=.1105
        P(17)=1
        Y(1,17)=.622
        Y(2,17)=.1322
        P(23)=1
        Y(1,23)=.559
        Y(2,23)=.1405
        P(34)=1
        Y(1,34)=.499
        Y(2,34)=.1468
        P(40)=1
        Y(1,40)=.482
        Y(2,40)=.1477
        P(45)=1
        Y(1,45)=.471
        Y(2,45)=.1477
        P(170)=1
        Y(1,170)=.443
        Y(2,170)=.1476
C
C       INTRODUZINDO K1,K2
```

```
        CK1=.242
        CK2=.428
C       FIM DA INICIALIZACAO
        DO 10 IRREST=1,2
        CALL GRAMI
        WRITE(ID,101)U(1),U(2)
101     FORMAT('U1= ',G20.8,'U2= ',G20.8)
        DO 20 I=1,NPT
20      WRITE(ID,100)I,X(1,I),X(2,I),X(3,I)
10      CONTINUE
100     FORMAT(2X,4G20.8)
        END
C
C       FUNX
        SUBROUTINE FUNX(T,H,F,UDT)
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
        COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        COMMON A(4),B(4),C(4),N11
C
C       COMMONS DO USUARIO
C
        COMMON P(201),Y(2,201),CK1,CK2
        DIMENSION UDT(201),F(11)
C         TYPE 334,X1(1),X1(2),X1(3)
334       FORMAT('X1(1),X1(2),X1(3) = ',3G)
          R1=UDT(1)*(X1(1)**2-X1(2)*(2.-2.*X1(1)-X1(2))/(3.*CK1))
          R2=UDT(2)*(X1(1)*X1(2)-(1.-X1(1)-2*X1(2))*(2-2*X1(1)-X1(2))
     1/(9*CK2))
C         TYPE 335,R1,R2
335       FORMAT( ' R1,R2 = ',2G)
          F(1)=-(R1+R2)
          F(2)=R1/2-R2
          IX=(T-TI)
          I=IX/DT+1
C         TYPE 111,F(3)
111       FORMAT(' F3= ',G)
          F(3)=P(I)*((Y(1,I)-X1(1))**2+(Y(2,I)-X1(2))**2)
        RETURN
        END
C
C       FUNL
        SUBROUTINE FUNL(T,H,F,UDT)
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
        COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        COMMON A(4),B(4),C(4),N11
C
C       COMMONS DO USUARIO
C
        COMMON P(201),Y(2,201),CK1,CK2
C
        DIMENSION UDT(201),F(11)
        I=(T-TI)/DT+1
          Z1=X(1,I)+X(2,I)/(3*CK1)
          Z2=X(2,I)+(4*(1-X(1,I))-5*X(2,I))/(9*CK2)
          Z3=(X(1,I)+X(2,I)-1)/(3*CK1)
          Z4=X(1,I)+(5*(1-X(1,I))-4*X(2,I))/(9*CK2)
          S11=-2*UDT(1)*Z1-UDT(2)*Z2
          S12=-2*UDT(1)*Z3-UDT(2)*Z4
```

```
      S21=UDT(1)*Z1-UDT(2)*Z2
      S22=UDT(1)*Z3-UDT(2)*Z4
      S31=-2*P(I)*(Y(1,I)-X(1,I))
      S32=-2*P(I)*(Y(2,I)-X(2,I))
      F(1)=-(S11*X1(1)+S21*X1(2)+S31*X1(3))
      F(2)=-(S12*X1(1)+S22*X1(2)+S32*X1(3))
      F(3)=0.
    RETURN
    END
C
C   GRAXF
    SUBROUTINE GRAXF
    COMMON U(201),G(201),X(10,201),ALBDA(10,201)
    COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
    COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
    COMMON A(4),B(4),C(4),N11
C
C   COMMONS DO USUARIO
C
    COMMON P(201),Y(2,201),CK1,CK2
C
      X1(1)=0.
      X1(2)=0.
      X1(3)=1.
    RETURN
    END
C
C   GRAD1
    SUBROUTINE GRAD1(GA,I,J)
    COMMON U(201),G(201),X(10,201),ALBDA(10,201)
    COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
    COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
    COMMON A(4),B(4),C(4),N11
C
C   COMMONS DO USUARIO
C
    COMMON P(201),Y(2,201),CK1,CK2
C
      Z1=X(1,I)**2-X(2,I)*(2-2*X(1,I)-X(2,I))/(3*CK1)
      Z2=X(1,I)*X(2,I)-(1-X(1,I)-2*X(2,I))*(2-2*X(1,I)-X(2,I))/(9*C
    IF(1-J)11,10,10
10    GA=-Z1*ALBDA(1,I)-Z2*ALBDA(2,I)
    GO TO 30
11    GA=-Z2*ALBDA(1,I)+Z1/2*ALBDA(2,I)
30  CONTINUE
    RETURN
    END
C   FI
    SUBROUTINE FI(AC)
    COMMON U(201),G(201),X(10,201),ALBDA(10,201)
    COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
    COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
    COMMON A(4),B(4),C(4),N11
C
C   COMMONS DO USUARIO
C
    COMMON P(201),Y(2,201),CK1,CK2
C
      AC=X1(3)
    RETURN
    END
```

APÊNDICE A8: LISTAGEM DO GRAMIN

```
C       GRAMIN = MINIMIZACAO PELO GRADIENTE CONJUGADO
C
        SUBROUTINE GRAMI
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
        COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        DIMENSION ALFO(201)
C
        CALL INIT I
        CALL GRAD
        NP=NPTOSG+1
        NPT=NPTOS+1
        DO 10 I=1,NP
10      SI1(I)=-1.*G(I)
        DO 20 I=1,NITER
20      ALFO(I)=0
        NALFO=1.
30      CALL BUSCA(ALFA,ALFO,NALFO)
        ALFO(NALFO)=ALFA
        DO 40 I=1,NP
40      U(I)=U(I)+ALFA*SI1(I)
         CALL PAGI(CT1,G,NP)
        CALL ESCSU(G,G,MODE,ESC,NPTOSG,DT)
        ESC1=ESC
        CALL GRAD
        CALL PAGI(CT1,G,NP)
        CALL ESCSU(CT1,CT1,MODE,ESC,NPTOSG,DT)
        CALL CUSTO (U,ACPRT)
        WRITE(ID,200) NALFO,ESC,ACPRT
200     FORMAT('NALFO= ',G14.8,'ESC= ',G14.8,'CUSTO= ',G14.8)
        IF(ESC1)81,80,81
   80   ESC1=1E-30
   81   BETA=ESC/ESC1
        DO 50 I=1,NP
50      SI1(I)=BETA*SI1(I)-G(I)
        NALFO=NALFO+1
379     FORMAT(2X,5G20.8)
C       DO 377 I=1,NPT
C377     WRITE(ID,378)I,X(1,I),X(2,I),X(3,I),X(4,I),U(I)
378     FORMAT(2X,6G20.8)
        IF(NITER-NALFO)60,30,30
60      RETURN
        END
C
        SUBROUTINE RUNGE(T,H,XOUL,UDT)
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
        COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        COMMON A(4),B(4),C(4),N11
        DIMENSION F(11),UDT(201),Y(12),AK(12)
        EQUIVALENCE(Y(2),X1(1)),(AK(2),F(1))
        Y(1)= T
        AK(1)=1.
        DO 10 J=1,4
        IF(XOUL)7,7,8
C       XOUL=0 PARA FUNX
C       XOUL=1 PARA FUNL
7       CALL FUNX(T,H,F,UDT)
        GO TO 9
```

```
 8      CALL FUNL(T,A,F,ODT)
 9      DO 10 I=1,N11
        Z=A(J)*(AK(I)-B(J)*Q(I))
        Y(I)=Y(I)+H*Z
10      Q(I)=Q(I)+3*Z-C(J)*AK(I)
        RETURN
        END
C       CALCULO DO GRADIENTE
C
        SUBROUTINE GRAD
C
C       MODO=1 := ESPACO DE FUNCOES CONTINUAS
C       MODO=2 := ESPACO DOS PARAMETROS
C       MODO=3 := ESPACO DE ENTRADAS AMOSTRADAS
C       MODO=4 := ESPACO DE M.L.P.
C       MODO=5 := ESPACO DO USUARIO
C
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CI1(201),SI1(201),IPGK,ALP,'D
        COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        DIMENSION OL(201)
C
10      Q(N+1)=0
        DO 20 I=1,N
        Q(I)=0
20      X1(I)=X0(I)
        H=DT
        T=TI
        NP=NPTOSG+1
        NPT=NPTOS+1
        IF(MODO-4)11,12,11
12      DO 26 I=1,NPT
        CALL USUB(OL,T,U)
26      T=T+H
        T=TI
11      DO 30 I=1,NPT
        IF(MODO-4)27,28,27
28      CALL RUNGE(T,H,0.,OL)
        GO TO 29
27      CALL RUNGE(T,H,0.,U)
29      DO 25 J=1,N
25      X(J,I)=X1(J)
30      T=T+H
        CALL GRAXF
        N11=N+1
        DO 40 I=1,N11
40      Q(I)=0
        H=-DT
        T=T+H
        DO 70 I=1,NPT
        IF(MODO-4)31,32,31
32      CALL RUNGE(T,H,1.,OL)
        GO TO 33
31      CALL RUNGE(T,H,1.,U)
33      DO 60 J=1,N
        ALBDA(J,NPTOS+2-I)=X1(J)
60      CONTINUE
70      T=T+H
        GO TO(80,100,130,160,210),MODO
```

```
C       ESPAÇO DE FUNÇÕES CONTINUAS
C
80      DO 90 I=1,NP
        I1=I
        CALL GRADI(GA,I1,0)
90      G(I)=GA
        RETURN
C
C       ESPACO DOS PARAMETROS
C
100     DO 120 J=1,NP
        DO 110 I=1,NPT
        J1=J
        I2=I
        CALL GRADI(GA,I2,J1)
110     CT(I)=GA
        CALL SYMP(CT,ESC,NPTOS,DT)
120     G(J)=ESC
        RETURN
C
C       ESPACO DE ENTRADAS AMOSTRADAS
C
130     NG=1
        NTE=NPTOS/M
        NPX=NPTOS-NTE+1
        DO 150 I=1,NPX,NTE
        I1=I
        DO 140 J=1,NTE
        CALL GRADI(GA,I1,NG)
        I1=I+J
140     CT(J)=GA
C       DEVE INTEGRAR PARA UM NUMERO INPAR DE PTOS.
        J=J+1
        CALL GRADI(GA,I1,NG)
        CT(J)=GA
        CALL SYMP(CT,ESC,NTE,DT)
        G(NG)=ESC
150     NG=NG+1
        RETURN
C
C       ESPACO DE MODULACAO POR LARGURA DE PULSO (M.L.P.)
160     NTE=NPTOS/M
        DO 200 J=1,M
        IF(ABS(U(J))-1)212,211,211
211     AUX=1.
        GO TO 213
212     AUX=U(J)
213     TP=(J-1)*NTE+ABS(AUX)*NTE
        I2=TP+1
        IF(U(J))204,201,204
204     IF(ABS(U(J))+U(J))201,202,201
201     I3=1
        GO TO 203
202     I3=-1
C       I2 FORNECE O INDICE DE ALBDA
C       I3 E A FUNCAO SIGN
203     CALL GRADI(GA,I2,I3)
        G(J)=GA*NTE
200     CONTINUE
        RETURN
```

```
C
C       ESPACO DO USUARIO
C
210     CALL GRADI
        RETURN
        END
C
C       ESCSUB=PRODUTO ESCALAR
C       MODE=0 PROD. ESC. DE VETORES
C       MODE=1 PROD. ESC. DE FUNCOES
C
        SUBROUTINE ESCSUB(CB1,CB,MODE,ESC,NDIM,DX)
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
        DIMENSION CB(201),CB1(201),CB2(201)
        ESC=0
        NDI=NDIM+1
        IF(MODE)10,10,20
10      DO 30 I=1,NDI
30      ESC=ESC+CB(I)*CB1(I)
        RETURN
20      DO 40 I=1,NDI
40      CB2(I)=CB(I)*CB1(I)
        CALL SYMP(CB2,ESC,NDIM,DX)
        RETURN
        END
C
C       SYMP INTEGRACAO PELA REGRA DE SYMPSON
C
C       O NUMERO TOTAL DE PTOS. E IMPAR, N E PAR
C
        SUBROUTINE SYMP(Y,SY,N,DX)
        DIMENSION Y(201)
C       DX=INTERVALO ENTRE CADA PTO.
C       N=NUMERO TOTAL DE PTOS-1
        N11=N+1
        SY=Y(1)
        DO 10 I=3,N11,2
10      SY=SY+4*Y(I-1)+2*Y(I)
        SY=SY-Y(N+1)
        SY=SY*DX/3.
        RETURN
        END
C
C       BUSCA
C       NALFO CONTA QUANTAS VEZES E CHAMADA A S/R
C       DURANTE AS ITERACOES
C       SE NALFO=1 E A PRIMEIRA VEZ QUE SE FAZ A BUSCA
C
C
        SUBROUTINE BUSCA(ALFA,ALFO,NALFO)
        COMMON U(201),G(201),X(10,201),ALBDA(10,201)
        COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
        COMMON XO(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
        DIMENSION ALFO(201)
        NP=NPTOSG+1
        NPI=NPTOS+1
        CALL CUSTO(U,A1)
40      IF(NALFO-1)60,50,60
```

```
 50     CALL ESCSU(G,G,NODE,ESC,NPTOSG,DT)
        B=.03*A1/ESC
C       B=B/1024.
        NSW=10
        WRITE(ID,300)B
300     FORMAT(' BUSCA B=',G16.8)
        GO TO 80
 60     B=1
        ND=NALFO-1
        DO 70 I=1,ND
        WRITE(ID,301)I,ALFO(I)
301     FORMAT(' BUSCA I=',I4,' ALFOI=',G16.8)
        IF(ALFO(I))70,71,70
 71     ALFO(I)=1.E-30
 70     B=B*ALFO(I)**(1./ND)
C       B=B/8.
        NSW=3
 80     DO 90 I=1,NP
 90     CT(I)=U(I)+B*SI1(I)
        CALL CUSTO(CT,A2)
        WRITE(ID,302)A1,A2
302     FORMAT(' BUSCA A1=',G16.8,' A2=',G16.8)
        IF(A1-A2)91,92,92
 91     KM=-1
        GO TO 110
 92     KM=1
110     K=0
        AD=1
        JCON=1
114     IF(AD)117,117,115
115     AA=2.**(KM*K)*B
        AB=2.**(KM*(K+1))*B
117     AC=2.**(KM*(K+2))*B
        IF(AD)135,135,118
118     DO 120 I=1,NP
120     CT(I)=U(I)+AA*SI1(I)
        CALL CUSTO(CT,AAT)
        DO 130 I=1,NP
130     CT(I)=U(I)+AB*SI1(I)
        CALL CUSTO(CT,ABT)
135     DO 140 I=1,NP
140     CT(I)=U(I)+AC*SI1(I)
        CALL CUSTO(CT,ACT)
        WRITE(ID,303)AA,AB,AC,AAT,ABT,ACT
303     FORMAT(' BUSCA AA=',G14.8,' AB=' ,G14.8,' AC=',G14.8,'AAT=',
      1G14.8,' ABT=',G14.8,' ACT=',G14.8)
        IF(AAT-ABT)170,150,150
150     IF(ABT-ACT)170,160,160
160     IF(A1-ACT)161,295,161
295     JCON=JCON+1
        IF(JCON-NSW)161,161,200
161     K=K+1
        AA=AB
        AB=AC
        AAT=ABT
        ABT=ACT
        AD=0
        GO TO 114
170     IF(AA-AB)501,501,502
502     ACOP=AA
```

```
      ACOPT=AAT
      AA=AC
      AAT=ACT
      AC=ACOPT
      ACT=ACOPT
  501 DO 176 ICON=1,4
      DDK=.5*(AA+AB)
      DEK=.5*(AB+AC)
      DO 171 I=1,NP
  171 CT(I)=U(I)+DDK*SI1(I)
      CALL CUSTO(CT,DDKT)
      DO 172 I=1,NP
  172 CT(I)=U(I)+DEK*SI1(I)
      CALL CUSTO(CT,DEKT)
      WRITE(ID,304)DDK,DEK,DDKT,DEKT
  304 FORMAT(' BUSCA DDK=',G14.8,' DEK=',G14.8,' DDKT',G14.8,
     1' DEKT=',G14.8)
      IF(AAT-DDKT)175,174,174
  174 IF(DDKT-ABT)175,175,250
  175 AC=AB
      ACT=ABT
      AB=DDK
      ABT=DDKT
      GO TO 176
  250 IF(ABT-DEKT)260,260,255
  260 AA=DDK
      AAT=DDKT
      AC=DEK
      ACT=DEKT
      GO TO 176
  255 AA=AB
      AAT=ABT
      AB=DEK
      ABT=DEKT
  176 CONTINUE
      E1=AAT*(AC**2-AB**2)+ABT*(AA**2-AC**2)
     1+ACT*(AB**2-AA**2)
      E2=AAT*(AC-AB)+ABT*(AA-AC)+ACT*(AB-AA)
      WRITE(ID,305)E1,E2
  305 FORMAT(' BUSCA E1=',G16.8,' E2=',G16.8)
      IF(E2)181,182,181
  182 E2=1.E-30
  181 DD1=0.5*E1/E2
      DO 180 I=1,NP
  180 CT(I)=U(I)+DD1*SI1(I)
      CALL CUSTO(CT,DD1T)
      WRITE(ID,306)AB,DD1,ABT,DD1T
  306 FORMAT(' BUSCA AB=',G16.8,' DD1=',G16.8,' ABT=',G16.8,' DD1T=',
     1G16.8)
      IF(ABT-DD1T)200,200,190
  190 ALFA=DD1
      RETURN
  200 ALFA=AB
      RETURN
      END
C
      SUBROUTINE CUSTO(BT,AC)
C     BT=VETOR DE ENTRADA DE NPTOSG COMPONENTES
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
```

```
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      DIMENSION UT(201),UL(201)
      NPI=NPTOS+1
      Q(N+1)=0.
      DO 10 I=1,N
      Q(I)=0
10    X1(I)=X0(I)
      H=(TF-TI)/NPTOS
C     NPTOS=NUMERO DE INTERVALOS ENTRE TI E TF,E NAO DE PTOS.
      T=TI
      IF(MODO-4)50,12,50
12    DO 26 I=1,NPT
      CALL USUB(UL,T,UT)
26    T=I+H
      T=TI
50    DO 30 I=1,NPI
       IF(MODO-4)29,28,29
28    CALL RUNGE(T,H,0.,UL)
      GO TO 30
29    IF(IPGR-1)27,200,27
200   CALL PAG(UT,T,UL)
      GO TO 28
27    CALL RUNGE(T,H,0.,UT)
30    T=I+H
C
C     X E Q SAO AUTOMATICAMENTE INCREMENTADOS
C     PELA S/R RUNGE
C
      CALL FI(AC)
C
      RETURN
      END
C
      SUBROUTINE INITI
      COMMON U(201),S(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),SI1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),N11
      A(1)=.5
C       IPGK=0
      B(1)=2.
      B(2)=1.
      B(3)=1.
      B(4)=2.
      C(1)=.5
      C(4)=.5
      N11=N+1
      A(2)=1.-SQRT(.5)
      A(3)=1.+SQRT(.5)
      A(4)=1./6.
      C(2)=A(2)
      C(3)=A(3)
      DT=(TF-TI)/NPTOS
      IF(MODO-4)10,20,10
C20     IPGK=0
20    ALP=1.
10    CONTINUE
        RETURN
        END
      SUBROUTINE USUB(ULA,T,UDT)
```

```
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON M,N,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
      COMMON A(4),B(4),C(4),NII
      DIMENSION UDT(201),ULA(201)
      NTE=NPTOS/M
      I=(T-TI)/DT
      K=I/NTE+1
      AUX=UDT(K)
      I1=I+1
      IF(UDT(K))61,21,61
61    IF(ABS(UDT(K))-1)50,60,60
60    AUX=1.
        UDT(K)=UDT(K)/ABS(UDT(K))
50    IF((ABS(AUX)*NTE)+(K-1)*NTE-I)21,21,10
10    IF(ABS(UDT(K))+UDT(K))30,40,30
30    ULA(I1)=1.
      GO TO 20
40    ULA(I1)=-1
       GO TO 20
21    ULA(I1)=0.
20    CONTINUE
      RETURN
      END
      SUBROUTINE PAGI(P1,P2,NP)
      COMMON U(201),G(201),X(10,201),ALBDA(10,201)
      COMMON N,M,NPTOS,NITER,CT(201),CT1(201),ST1(201),IPGK,ALP,ID
      COMMON X0(10),TI,TF,DT,NPTOSG,MODE,MODO,MX,X1(11),Q(11)
       DIMENSION P1(201),P2(201)
       IF(IPGK-1)80,20,80
20    DO 50 I=1,NP
      IF(ABS(U(I))-ALP)30,40,40
30    P1(I)=P2(I)
      GO TO 50
40    P1(I)=0.
50    CONTINUE
      GO TO 70
80    DO 60 I=1,NP
60    P1(I)=P2(I)
70    CONTINUE
      RETURN
      END
      SUBROUTINE PAG(BT,T,UL)
      DIMENSION BT(201),UL(201)
      RETURN
      END
```

B I B L I O G R A F I A

1- D'Azzo, J. J. & Houpis, C. H. ; " Linear Control Systems Analysis and Design" (McGraw-Hill Inc. 1975).

2- Coughanouwr, D. R. & Koppel, L. B. " Análise e Controle de Processos " (Guanabara Dois S.A. 1978 - tradução).

3- Bottura, C. P. ; "Princípios de controle e Servomecanismos", (ed:UNICAMP, 1980).

4- Tabak, D. & Kuo, B. C. ;" Optimal Control By Mathematical Programming" (Prentice-Hall, Englewood Cliffs, N. J. 1971).

5- Noton, M. ; "Modern Control Engineering" (Pergamon Press Inc , New York 1972).

6- Sage, A. P. ; "Optimum Systems Control" (Prentice-Hall Inc. , Englewood Cliffs, N. J. 1968).

7- Athans, M. & Falb, P. L. "Optimal Control: an Introduction to the Theory and its Aplications" (McGraw-Hill, New York, 1966).

8- Gelfand, I. M. & Fomin, S. V. ; "Calculus of Variations" (Prentice-Hall Inc. , Englewood Cliffs, N. J. 1963).

9- Arfken , G. ; "Mathematical Methods for Physicists" (Academic Press Inc., New York 1971).

10- Pontryagin L. S.,& Boltyanskii, V.G. & Gamkrelidze, R. V. , Mishchenko, E. F. ; "Mathematical Theory of Optimal Processes" (John Wiley and Sons 1962, tradução).

11- Bellman R. & Kalaba R.; "Dynamic Programming and Adaptive Processes: Mathematical Foundation" - IRE Transactions on Automatic Control, Vol. AC-5, 1960, pp. 5-10.

12- Bellman, R.E. ;"Dynamic Programming" (Princenton University Press, Princenton, N.J. 1972).

13- Ritch, P.S. ; "Discrete Optimal Control with Multiple Constraints I: Constraints Separation and Transformation Techique" - Automatica, Vol. 9, pp. 415-429 , Pergamon Press 1973.

14- Larson, R.E.; "State Increment Dynamic Programming" (American Elseviev 1968).

15- Sakarovitch, M. ; "Notes on Linear Programming" (Van Nostrand Reinhold, New York 1970).

16- Puccini, A.L. ; "Introdução à Programação Linear" (Livros técnicos e Científicos Editora S.A., 1980).

17- Lasdon,L.S. ; "Optimization Theory for Large Systems" (MacMillan, New York; Collier-MacMillan, London 1970).

18- Zoutendijk, G.;"Mathematical Programming Methods"(North-Holland Publishing Comp., New York 1976).

19- Sposit, V.A. ;"Linear and Nonlinear Programming" (The Iowa State University Press, 1975).

20- Dantizig, G.B. ; "Linear Programming and Extensions" (Princeton University Press, Princeton N.J., 1963).

21- Ralston A. & Wilf , H.S. , "Mathematical Methods for Digital Computers", vol. I (Jonh Wiley & Sons, Inc. 1976).

22- White, W.W. ; "A Status Report on Computing Algorithms for Mathematical Programming" , Computing Surveys, Vol. 5, Nº 3, September 1973 , pp. 135-166.

23- Wismer, D.A. & Chattergy R. ;"Introduction to Nonlinear Optimization: A Problem Solving Aproach (North-Holland Series in Systems Science and Engineering, 1978).

24- Varaya, P.P. ;"Notes on Optimization" (Van Nostrand 1972).

25- Molina, W.F. ;"A Survey of Resource Directive Decomposition in Mathematical Programming", Computing Surveys, Vol. 11, Nº 2, junho 1979 , pp. 95-104.

26- Luemberger, D.G. ;" Introduction to Linear and Nonlinear Programming" (Addison-Wesley 1973).

27- Lasdon , L.S.& Mitter, S.K. & Waren, A.D. ; "The Conjugate Gradient Method for Optimal Control Problems", IEEE Trans. on Automatic Control , AC-12 (2), pp. 132-138, abril 1967.

28- Hasdorff, L. ; "Gradiente Optimization and Nonlinear Control" (John Wiley & Sons, New York, 1976).

29- Schley, C.H. & Lee, I. ;"Optimal Control Computation by Newton -Raphson Method and Riccati Transformation", IEEE Trans. on Automatic Control, AC-12, Nº 2, pp. 139-144, abril 1967.

30- Lasdon, L.S. & Waren, A.D. & Rice, R.K. ;"An Interior Penalty Method for Inequality Constrained Optimal Control Problems" , IEEE Trans. on Automatic Control, AC-12, Nº 4, pp. 387-394, agosto 1967.

31- Bertsekas, D.P. ; "Partial Conjugate Gradient Methods for a Class of Optimal Control Problems", IEEE Trans. on Automatic Control, AC-19, Nº 3, pp. 209-217, junho 1974.

32- Fong, T.S. & Leonds, C.T.; "Method of Conjugate Gradients for Optimal Control Problems with State Variable Constraint"-Advances in Control Systems vol. 8,(New York Academic Press, 1971).

33- Lasdon L.S.; "Conjugate Direction Methods for Optimal Control", IEEE Trans. on Automatic Control, AC-15, Nº 2, pp. 267-268, abril 1970.

34- Abadie, J. ;"Non Linear and Integer Programming" (North Holland Publishing , 1978),capítulo 8.

35- Tripathi, S.S. & Narendra, K.S.; "Optimization Using Conjugate Gradient Methods", IEEE Trans. on Automatic Control, AC-3, Nº 1 pp. 268-270, abril 1970.

36- Aoki, M. ;"Introduction to Optimization Techiniques:Fundamentals and Application of Nonlinear Programming." (The Macmillam Company, New York, 1971).

37- Himmelblau D.M. ;" Applied Nonlinear Programming" (McGraw-Hill Company 1972).

38- Fletcher,R.;"A Review of Methods for Unconstrained Optimization" - do livro: "Optimization" (ed:Fletcher; London and New York Academic Press, 1969).

39- Powell, M.J.D. ;"A Survey of Numerical Methodos for Unconstrained Optimization", SIAM Review Vol. 12, Nº 1,pp. 79-97, janeiro 1970.

40- Rosembrock, H.H. ;"An Automatic Method for Finding the Greasted or Least value of a function", The Computer Journal, Vol 3, Nº 3, pp. 175-184, outubro 1970.

41- Powell, M.J. D.;"An efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives" The Computer Journal, Vol. 7, Nº 2, pp. 155-162, julho 1964.

42- Zangwill , W.I.; "Minimizing a Function without Calculating Derivatives", The Computer Journal, Vol. 10, pp. 293-296, novembro 1967.

43- Cauchy, A. ; "Method Générale pour la Resolution des Systemes D'Equations Simultnées", C. R. Acad. Sci. Paris, 25 (1847), p. 536

44- Fletcher, R. & Reeves, C.M. ;"Function Minimization by Conjugate Gradients", The Computer Journal,Vol. 7, Nº 7, pp.149-153.

45- Box, M.J. ;" A Comparison of Several Current Optimization Methods,and the Use of Transformations in Constraineds Problems" The Computer Journal, Vol. 9, pp. 67-77, maio 1966.

46- Davidon, W. C. ; "Variable Metric Method for Minimization", A.E.C. Research and Development Report, ANL-5990(Rev.) 1959.

47- Fletcher, R. & Powell M.J.D. ; "A Rapidily Convergent Descent Method for Minimization", The Computer Journal, Vol. 6, pp.163 -168, 1963.

48- Hestenes, M.R. & Stiefel, E. ;"Methods of Conjugate Gradients for Solving Linear Systems", J. Res. N.B.S. Vol 49, pp. 409, (1952).

49- Pagurek, B. & Woodside, C.M. ; "The Conjugate Gradient Method for Optimal Control Problems with Bounded Control Variables", Automatica, Vol 4, pp. 337-349, 1968.

50- Luemberger, D.G. ; "Optimization by Vector Space Methods", (John Wiley & Sons, Inc. , New York - 1969).

51- Dieudonné, J. "Foundations of Modern Analysis" (Academic Press - 1962).

52- Chen, C.T. "Introduction to Linear System Theory" - (Holt Rinehart and Wiston, Inc. - New York - 1970).

53- Jacobson, D.H. & Lele, M.M. ; "A Transformation Technique for Optimal Control Problems with a State Variable Inequality Constraint", IEEE Trans. on Automatic Control, Vol.AC-14, Nº 5, pp. 457-464, outubro 1969.

54- Miele, A. & Wu, A.K. & Liu, C.T. ;"A Transformation Technique for Optimal Control Problems with Partially Linear State Inequality Constraints"; Journal of Optimization Theory and Applications, Vol 28, Nº 2, pp. 185-212, junho 1979.

55- Athans, M. & Dertouzos M.L. & Spann R.N. & Mason S.J.;"Systems Networks, and Computation Multivariables Methods "(McGraw-Hill, Inc - 1974).

56- McCracken, D.D. & Dorn, W.S.; "Numerical Methods and Fortran Programming" (Jonh Wiley and Sons Inc. , New York, 1964).

57- Lápidus, Seinfeld; " Mathematical Methods for Chemical Engineering " , vol. 3 .

58- Levenspiel, Octave; " Ingenieria de Las Reacciones Quîmicas", (Reverté SA; 1974).

59- " The Apple Tutorial" (Apple Computer INc. 1979).

60- " NE-Z8000 , Manual de Intruções"(Editele; Editora Tecnica Eletronica LTDA).

61- Zuffo, J.A. ;" Fundamentos da Arquitetura e Organização dos microcomputadores" (Edgard Blucher , 1978).

62- Hogan, T. "Discover Forth" (Osborne/ MacGraw-Hill , 1982)