Capítulo 4

ALGORITMO DE OTIMIZAÇÃO

4.1 – INTRODUÇÃO

Neste capítulo, discute-se o algoritmo de otimização utilizado para o projeto inverso de aerofólios em grades de turbomáquinas. É importante salientar que se trata de uma abordagem bastante diferente daquela desenvolvida anteriormente na UNIFEI, que se baseava no próprio método dos painéis, tratava todos os pontos nodais como variáveis de projeto e não utilizava algoritmos de otimização explícitos (Petrucci, 2003). A vantagem da abordagem apresentada neste trabalho é que ela independe do método de cálculo do escoamento e do tipo de parametrização empregados, e até mesmo do algoritmo de otimização. Contudo, neste trabalho emprega-se o método dos painéis descrito no Capítulo 2, a técnica de parametrização descrita no Capítulo 3 e um algoritmo de otimização baseado em técnicas populacionais evolucionárias.

Mesmo com técnicas populacionais de otimização mais consagradas em aerodinâmica, como os Algoritmos Genéticos (GA) ou de Evolução Diferencial (DE), optou-se neste trabalho pelo algoritmo de Busca Aleatória Controlada (CRS), proposto inicialmente por

Price (1977) e desenvolvido em novas versões por Ali et.al. (1997*a*). Tal escolha justifica-se por se tratar de um algoritmo de fácil implementação e com bom potencial de aplicabilidade, além de poder atingir um desempenho comparável ao dos GA´s e DE´s. Outro fator considerado é a aparente escassez de trabalhos de aplicação dos algoritmos CRS em problemas inversos e de otimização de aerofólios ou pás de turbomáquinas. Assim, o presente trabalho busca contribuir no sentido de avaliar as potencialidades dessa opção adicional. Busca também contribuir com a apresentação de uma nova versão do CRS, mais eficiente para o problema de projeto inverso de aerofólios (Manzanares Filho *et al.*, 2005).

4.2 – ALGORITMOS DE BUSCA ALEATÓRIA CONTROLADA

Os algoritmos de busca aleatória controlada são algoritmos de otimização adequados para a pesquisa de minimizadores globais de uma função real contínua, denominada função objetivo, f: $\Re^n \rightarrow \Re$, não necessariamente diferenciável, definida em uma hiper-caixa $S = \{\mathbf{x} \in \Re^n\text{: } x_j^L \leq x_j \leq x_j^U, j = 1, ..., n\}$, onde x_j^L e x_j^U representam, respectivamente, limites inferiores e superiores para as n coordenadas de $\mathbf{x}$. Um ponto $\mathbf{x}^*$ é dito um otimizador global de f se $f(\mathbf{x}^*) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in S$. Além das restrições laterais usadas na definição de S, outros tipos de restrição poderiam, em princípio, ser impostos por meio de esquemas de penalidade sobre a função objetivo ou qualquer outra técnica de tratamento de restrições (Oyama *et al.*, 2005).

Os algoritmos CRS foram propostos como um aperfeiçoamento de métodos simples de busca aleatória em que apenas o ponto de menor valor da função objetivo fica retido em cada iteração (Price, 1977). Assim como os GA´s e DE´s, um CRS é um algoritmo populacional evolucionário que parte de um conjunto inicial P de N pontos distribuídos aleatoriamente sobre S e, a partir daí, efetua um processo de contração iterativa em direção a um ponto de mínimo global por meio de procedimentos puramente heurísticos (Ali *et al.*, 1997*a*, e Ali e Törn, 2004). Além disso, o tamanho N da população é mantido durante todo processo de otimização. Diferentemente dos demais algoritmos evolucionários mencionados, um CRS substitui um único ponto da população (seu ponto atual de pior valor da função objetivo, **h**) por um ponto melhor, **p**, em cada iteração (i.e., um ponto tentativo **p** tal que $f(\mathbf{p}) < f(\mathbf{h})$). Assim, sua implementação é mais direta

4.2.1 – O Algoritmo CRS Básico

O algoritmo CRS básico para minimização pode ser descrito brevemente do seguinte modo (adaptado de Ali *et al.*, 1997*a* e Ali e Törn, 2004):

1. Gerar aleatoriamente uma população inicial P de N pontos em S: $P = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. Calcular os valores da função nesses pontos de um modo indexado. Determinar o pior ponto, **h**, e o melhor ponto, **l**, i.e., aqueles pontos em P com o maior e o menor valor da função, f_h e f_l, respectivamente. Se um critério de parada já for atendido, parar (por exemplo, parar se $f_h - f_l < \epsilon$, onde ϵ é uma tolerância especificada). Se não, continuar.
2. Gerar um ponto tentativo **p** para substituir o pior ponto, **h**.
3. Se **p** for inviável ($\mathbf{p} \notin S$), ir ao passo 2 (ou, tornar **p** viável e continuar).
4. Avaliar $f_p = f(\mathbf{p})$. Se **p** for insatisfatório ($f_p \geq f_h$), ir ao passo 2.
5. Atualizar o conjunto P substituindo o pior ponto atual pelo ponto tentativo: ($P \leftarrow P \cup \{\mathbf{p}\} / \{\mathbf{h}\}$). Encontrar **h** e f_h na nova população P. Se $f_p < f_l$, estabelecer **p**, f_p como os novos **l**, f_l, respectivamente.
6. Se um critério de parada for atendido, parar; se não, ir ao passo 2.

As duas principais diferenças dentre as versões disponíveis do CRS estão relacionadas (*i*) ao modo de geração do ponto tentativo (passo 2) e (*ii*) ao acesso opcional a uma fase de busca local sempre que o melhor ponto for o mais novo na população (quando $f_p < f_l$ no passo 5). Deve-se observar que todas as versões admitem que $N >> n$ (por exemplo, Ali *et al.* (1997*a*) sugerem empregar $N = 10(n + 1)$).

4.2.2 – Algumas Versões do CRS

O CRS proposto por Price (1977) foi aparentemente o primeiro a aparecer no formato acima. Ele não inclui uma fase local no passo 5. A geração do ponto tentativo no passo 2 é efetuada do seguinte modo: Escolher aleatoriamente $n + 1$ pontos distintos da população atual P: $\mathbf{r}_1, ..., \mathbf{r}_{n+1}$ (formando um simplex em $\Re^n$). Determinar o centróide **g** dos n primeiros pontos $\mathbf{r}_1, ..., \mathbf{r}_n$. Em seguida, determinar o ponto tentativo **p** pela reflexão do ponto remanescente $\mathbf{r}_{n+1}$ em relação ao centróide **g**, segundo o procedimento de Nelder e Mead (1965):

$$\mathbf{p} = 2\mathbf{g} - \mathbf{r}_{n+1} \tag{4.1}$$

Neste esquema, o cálculo do ponto **g** é puramente geométrico: ele não leva em conta informações a respeito do comportamento da função em torno de **g**. Em que pese sua objetividade, capacidade de exploração e pequenas demandas de armazenamento, o CRS de Price (1977) exibe uma tendência de baixa taxa de convergência – o que motivou estudos adicionais buscando seu aprimoramento.

Ali *et al.* (1997*a*) compararam algumas versões modificadas do CRS. Os autores enumeraram essas versões cronologicamente da seguinte maneira: o CRS1 — o algoritmo original de Price (1977) que se acabou de descrever; o CRS2, também proposto por Price (1983) fazendo um uso mais sofisticado do simplex na geração do ponto tentativo; o CRS3, também devido a Price (1987), em que foi incluída uma fase de busca local; o CRS4, uma modificação do CRS2 pela inclusão de buscas locais aleatórias em torno do melhor ponto atual através de distribuições β de probabilidade (Ali e Storey, 1995); o CRS5, excluído das comparações de Ali *et al*. (1997) por empregar uma busca local baseada em gradientes que não se enquadrava no escopo do trabalho (Ali e Storey, 1995); finalmente, o CRS6, proposto por Ali *et al*. (1997*a*), em que a fase de buscas locais do CRS4 (baseadas na distribuição β) é mantida e a fase global passa a utilizar um esquema de interpolação quadrática.

No esquema de interpolação quadrática do CRS6, três pontos distintos da população P atual são tomados: o melhor ponto, $\mathbf{r}_1 = \mathbf{l}$, e dois outros pontos, aleatoriamente, $\mathbf{r}_2$ e $\mathbf{r}_3$. Os valores respectivos da função objetivo são $f_1 = f(\mathbf{r}_1)$, $f_2 = f(\mathbf{r}_2)$ e $f_3 = f(\mathbf{r}_3)$. Variando j de 1 a n, constroem-se interpolações quadráticas usando as correspondentes coordenadas j daqueles três pontos, $r_{1j} = l_j$, r_{2j} e r_{3j}. A coordenada p_j do ponto tentativo é igualada ao ponto *extremo* da interpolação quadrática (Fig. 4.1, Eq. 4.2):

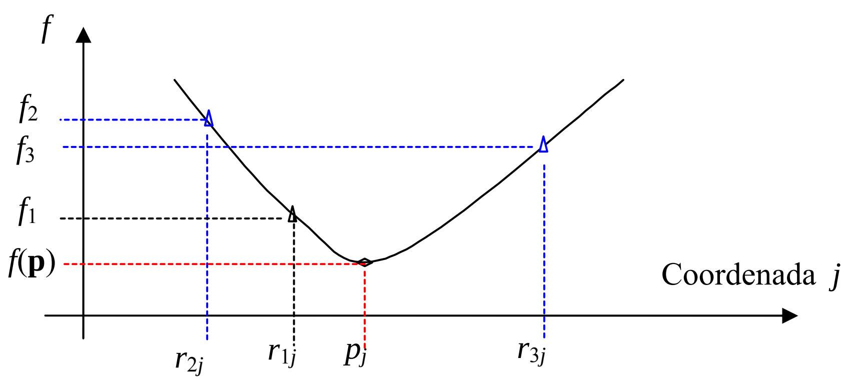


Figura 4.1 – Parábola para obtenção da coordenada j do novo ponto tentativo.

$$p_j = \frac{1}{2}\,\frac{(r_{2j}^2 - r_{3j}^2)\,f_1 + (r_{3j}^2 - r_{1j}^2)\,f_2 + (r_{1j}^2 - r_{2j}^2)\,f_3}{(r_{2j} - r_{3j})\,f_1 + (r_{3j} - r_{1j})\,f_2 + (r_{1j} - r_{2j})\,f_3}\ , \qquad (j = 1, ..., n) \qquad (4.2)$$

A idéia heurística subjacente a esse esquema consiste em considerar como promissora qualquer região em torno do melhor ponto e promover uma busca global por melhores pontos em tais regiões por meio de interpolações parabólicas ao longo de cada coordenada.

Ali *et al.* (1997*a*) compararam o CRS6 com outras versões do CRS em 13 problemas-teste consagrados em otimização global. Em todos os testes, o desempenho do CRS6 mostrou-se superior. Verificou-se que o esquema de interpolação quadrática foi mais importante para a melhoria do desempenho que as buscas locais baseadas na distribuição β de probabilidade — embora essas também tenham exercido um efeito benéfico.

4.2.3 – Um Algoritmo CRS Usando Reflexão Baseada em Variabilidade: CRS-VBR (Manzanares Filho *et al.*, 2005)

A facilidade de implementação e bom desempenho do CRS6 trazem uma expectativa a respeito de sua eficácia na solução de problemas de projeto de formas aerodinâmicas. Alguns testes preliminares foram realizados nesse sentido, concluindo-se que o CRS6 poderia, de fato, ser aplicado nesses problemas; entretanto, alguns inconvenientes também puderam ser verificados: para um número relativamente grande de parâmetros geométricos, como no caso da parametrização descrita no Capítulo 3 (n =17), há tendência de redução da taxa de convergência do CRS6 em direção a um mínimo global; as buscas locais baseadas na distribuição β de probabilidade nem sempre são benéficas e até podem dificultar a convergência; existem alguns parâmetros empíricos nessas buscas locais que são de difícil calibração — o que representa uma perda de objetividade e generalidade. Assim, torna-se desejável dispor de uma versão aprimorada do CRS capaz de eliminar ou, pelo menos, amenizar esses inconvenientes.

Após algumas análises críticas, foi possível identificar quatro aspectos do CRS6 merecedores de atenção:

(*i*) primeiramente, não é possível asseverar que o extremo da interpolação quadrática (Fig.4.1) corresponda sempre a um mínimo; situações de máximo podem ocorrer com relativa freqüência; (*ii*) as regiões de busca cobertas pelas interpolações quadráticas em torno do melhor ponto não possuem necessariamente a mesma potencialidade de fornecer pontos melhores — pode-se argumentar que regiões com fortes variações da função são mais promissoras que regiões com variações suaves; isso é especialmente importante quando a interpolação não produz um mínimo; (*iii*) podem ocorrer situações em que a interpolação torna-se mal-condicionada; (*iv*) finalmente, seria desejável eliminar as duvidosas buscas locais baseadas na distribuição β de probabilidade e a corresponde calibração de parâmetros empíricos.

Tendo em mente tais considerações, outra versão do CRS foi desenvolvida (Manzanares *et al.*, 2005). Essa versão — denominada CRS-VBR— faz uso seletivo das interpolações quadráticas do CRS6, leva em conta a variabilidade da função em torno do melhor ponto da população atual e elimina as buscas locais baseadas na distribuição β de probabilidade. As particularidades do CRS-VBR referem-se apenas à geração do ponto tentativo, i.e, ao passo 2 do CRS básico (item 4.2.1). Um valor médio da função, f_g, e uma medida de variabilidade local, α, são calculados da seguinte maneira:

$$f_g = \frac{1}{2}\left(f_2 + f_3\right), \tag{4.3}$$

$$\alpha = \frac{f_g - f_l}{f_h - f_l} \tag{4.4}$$

onde f_h representa o valor da função do pior ponto da população atual.

O algoritmo proposto para determinar cada coordenada p_i do ponto tentativo pode ser descrito da seguinte maneira:

1. Fazer um teste de mal-condicionamento da interpolação quadrática entre r_{1i}, r_{2i} e r_{3i}. *Se* ela for considerada mal-condicionada, *então* avaliar o valor de p_i aleatoriamente no intervalo viável correspondente, $x_i^L \leq x_i \leq x_i^U$.

2. *Se não*, *se* a coordenada do melhor ponto estiver entre as coordenadas dos outros dois pontos, — i.e., se $(r_{2i} - r_{1i})\ (r_{3i} - r_{1i}) < 0$ —, *então* calcular p_i como o mínimo da interpolação quadrática entre r_{1i}, r_{2i} e r_{3i}, assim como é feito no CRS6 (Eq. 4.2).

3. *Se não*, calcular uma coordenada centroidal ponderada g_i do seguinte modo:

$$g_i = \frac{(f_2 - f_1) r_{2i} + (f_3 - f_1) r_{3i}}{(f_2 - f_1) + (f_3 - f_1)} \tag{4.5}$$

e, em seguida, calcular a coordenada p_i do ponto tentativo por meio de uma reflexão baseada em variabilidade da coordenada centroidal g_i em relação à coordenada r_{1i} do melhor ponto:

$$p_i = (2 - \alpha) r_{1i} - (1 - \alpha) g_i \tag{4.6}$$

Observações

- A interpolação pode ser considerada mal-condicionada quando o determinante associado ao coeficiente do termo quadrático, $(r_{2i} - r_{3i})\, f_1 + (r_{3i} - r_{1i})\, f_2 + (\, r_{1i} - r_{2i})\, f_3$, ficar abaixo de um pequeno valor prescrito, δ_C. Opcionalmente, buscando maior generalidade, é possível usar o número de condição da matriz de interpolação, que é independente de escala, diferentemente do determinante. Neste trabalho, optou-se por usar $\delta_C = 10^{-30}$, adequado para dupla precisão. Além do tamanho N da população, δ_C torna-se o único parâmetro empírico adicional do algoritmo proposto.
- A avaliação aleatória de p_i quando a interpolação é considerada mal-condicionada incrementa a abrangência da busca e ajuda a evitar uma contração prematura da população em direção a um mínimo local. Mesmo uma pane eventual do processo numérico pode ser evitada desse modo.
- Ao satisfazer a condição de aceite de uma coordenada do ponto tentativo do CRS6, garante-se com certeza que essa coordenada corresponde ao mínimo da interpolação e, também, que ela permanece no intervalo viável.
- O cálculo da coordenada centroidal g_i (Eq. 4.5) coloca maior peso sobre a coordenada cujo valor da função é maior, a fim de enfatizar a variabilidade local em torno da coordenada do melhor ponto atual.
- A avaliação da coordenada do ponto tentativo (Eq. 4.6) utilizando a medida de variabilidade local α (Eqs. 4.4 e 4.3) representa um procedimento heurístico para balancear automaticamente buscas globais e locais. Para variabilidade baixa ($\alpha \to 0$), a coordenada centroidal é refletida para longe da coordenada do melhor ponto atual, pois o intervalo de interpolação correspondente não parece promissor nesse caso. Isso ajuda o algoritmo a escapar de pontos de mínimo local e torna a busca "mais global", quando

necessário. Para alta variabilidade ($\alpha \to 1$), a coordenada centroidal é refletida para perto da coordenada do melhor ponto atual, pois agora é de se esperar melhores valores da função por ali. Isso ajuda o algoritmo a acelerar a busca de um mínimo local, quando necessário.

4.3 – COMENTÁRIOS SOBRE A IMPLEMENTAÇÃO E UTILIZAÇÃO DO CÓDIGO COMPUTACIONAL

Os algoritmos CRS-VBR e CRS6 foram implementados em linguagem Fortran para utilizar a rotina `GRADLIN` descrita no Capítulo 2 como solucionador ("solver"), integrando assim a metodologia de projeto inverso de aerofólios proposta neste trabalho. Alguns aspectos da implementação merecem comentário: (1) dados de entrada; (2) geração da população inicial; (3) definição e cálculo da função objetivo; (4) critérios de parada.

4.3.1 – Dados de Entrada

Para utilizar o código computacional, devem-se fornecer os seguintes dados: (*i*) limites inferiores e superiores para as variáveis de projeto (Δ_i^L e Δ_i^U, i = 1, ..., n=17); (*ii*) tamanho da população de aerofólios N; (*iii*) semente do gerador de números aleatórios; (*iv*) número máximo de chamada da função objetivo; (*v*) tolerância para o critério de parada, ε; (*vi*) opção para usar CRS-VBR ou CRS6; (*vii*) no caso de se usar CRS6, opção para realizar ou não buscas locais baseadas em distribuições β; (*viii*) dados de grade linear: ângulo de montagem $\overline{\beta}$, razão de solidez ℓ/t e ângulo do escoamento na entrada α_1; (*ix*) número de painéis a ser usado na discretização dos aerofólios.

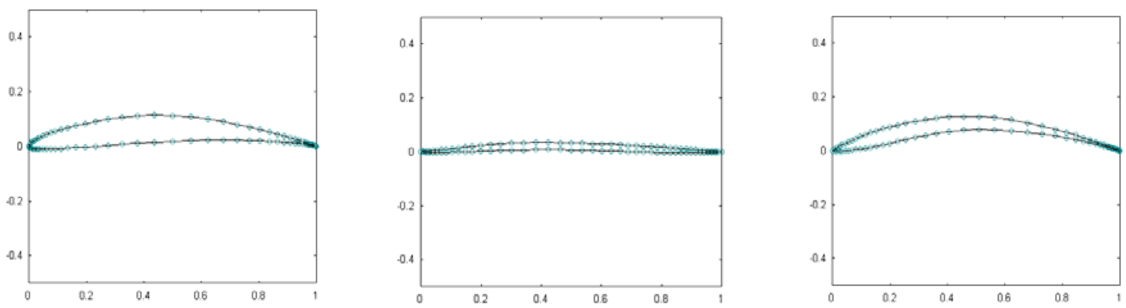
Na versão atual do código, os dados referentes aos itens (*i*), (*ii*) e (*ix*) acima estão pré-fixados no próprio código; os demais dados são fornecidos através da rotina `INPUT_PAR`.

4.3.2 – Geração da População Inicial

O tamanho da população atualmente pré-fixado no código computacional segue sugestão de Ali *et al.* (1997*a*): $N = 10\ (n+1)$. Portanto, com $n = 17$, tem-se $N = 180$. Esse valor pode ser redefinido no código e permanece constante durante o processo de otimização (pois haverá a substituição do pior perfil por um novo a cada iteração bem sucedida).

A geração da população inicial é feita pela rotina `INIT_POP`. Para cada aerofólio da população, ela chama a rotina `RAND_PAR`, que calcula aleatoriamente e armazena os valores das variáveis de projeto (Δ_i, $i = 1, ...,17$), as coordenadas da parametrização de Rogalsky ($b_i = 0, ..., 14$) e os respectivos valores da função objetivo. A transformação das variáveis Δ_i nas coordenadas b_i é feita conforme explicado na Tabela 3.4 do capítulo anterior. Para a população inicial, o valor da função objetivo é feito pela chamada da rotina `FCNR` de dentro da rotina `INIT_POP.` Para gerar os pontos tentativos durante o processo de otimização, a rotina `RAND_PAR` é chamada pelas rotinas de busca apenas para a viabilização do aerofólio e para a transformação $\Delta_i \rightarrow b_i$.

Assim, gerada a população inicial e conhecidas o melhor e o pior aerofólios, inicia-se o processo de busca aleatória controlada pela melhoria da população até que seja atingido o critério de parada. Exemplos de um perfil alvo e do pior e melhor perfis gerados na população inicial são mostrados na Figuras 4.2 .



(*a*) (*b*) (*c*)

Figura 4.2 – (*a*) Perfil alvo; (*b*) pior perfil gerado na população inicial; (c) melhor perfil gerado na população inicial

Note-se que, enquanto o pior perfil da população inicial está bastante distante do perfil alvo, o melhor perfil já apresenta melhor aproximação, por efeito exclusivo de uma geração aleatória. O processo de busca aleatória controlada deverá fazer todos os 180 perfis da população convergir para o perfil alvo.

4.3.3 – Definição e Cálculo da Função Objetivo

A função objetivo para o problema de projeto inverso é definida por uma norma, baseada na diferença entre a distribuição de velocidades ou pressões, calculada em cada iteração e aquela requerida para o aerofólio alvo (fixada). É possível introduzir ponderações adequadas a cada problema específico. No Capítulo 5, serão apresentados três casos de projeto inverso, cada qual com uma definição diferente dessa norma em virtude das características peculiares de cada caso.

O cálculo da função objetivo é feito pela chamada da rotina `FCNR` com os parâmetros da parametrização direta b_i (Rogalsky *et al.*, 1999) fornecidos. Essa rotina é chamada pela rotina `INIT_POP` na fase de geração da população inicial. Na geração do ponto tentativo, a rotina `FCNR` é chamada pelas rotinas de busca global e/ou local, `TRIAL_GLOBAL` e `TRIAL_LOCAL`, respectivamente.

Ao ser chamada, a rotina `FCNR` chama primeiramente a rotina `GEOM`, que realiza a discretização do aerofólio, usando os parâmetros b_i correspondentes. Em seguida, é chamada a rotina `GRADLIN` para calcular as distribuições de velocidades e pressões sobre o aerofólio. Finalmente, essas distribuições são comparadas com a distribuição alvo através do cálculo da função objetivo desejada.

4.3.4 – Critérios de Parada

O processo iterativo é interrompido quando um de dois critérios for primeiramente atendido: (*i*) o número de chamadas da função objetivo ultrapassar um número máximo especificado ou (*ii*) a diferença entre os valores da função objetivo do pior e do melhor aerofólio da população corrente ficar abaixo da tolerância pré-estabelecida: $f_h - f_l < \varepsilon$.

Um fluxograma do programa apresenta-se no Apêndice A