Projeto Inverso de Bocal Convergente-Divergente

por Allan Moreira de Carvalho
Universidade Federal do ABC

24 agosto 2022

1 Introdução

O projeto de superfícies aérodinâmicas é um problema recorrente da engenharia aeroespacial, em sua forma mais antiga, tais projetos eram realizados por meio de sucessivos experimentos à partir do qual propunham-se alterações na geometria à fim de obter-se o escoamento desejado. Tal processo tornar-se excessivamente oneroso quando a geometria em questão é muito complexa ou as condições do escoamento são extremas. Nesse sentido as ferramentas de fluidodinâmica computacional permitem que uma enorme quantidade de experimentos numéricos possam ser realizados, com um custo muito acessível. Além disso os métodos numéricos podem ser facilmente utilizados em projetos inversos.

O projeto inverso consiste em, à partir de determinada distribuição do escoamento, determinar-se qual a geometria que o produz. Nesse tipo de projeto os métodos de otimização desempenham papel central. Nesse trabalho, diferentes algoritmos e estratégias de otimização serão analisadas para a solução do problema inverso de um bocal convergente-divergente, inspirado no trabalho [1].

2 Objetivo

Os objetivos desse trabalho são:

- Aplicar os métodos de otimização determinísticos disponíveis na biblioteca `scipy` para a solução do problema inverso de projeto aerodinâmico de um bocal convergente-divergente.
- Avaliar a performance dos métodos em dois cenários, utilizando uma parametrização geométrica polinomial com apenas uma variável independente e em outro cenário utilizando uma *spline* cúbica.

3 Fundamentação Teórica

3.1 Método Numérico para solução do escoamento

O principal aspecto do escoamento em bocais convergente-divergentes é a formação de ondas de choque que podem ocorrer no regime supersônico, por esse motivo o modelo deve considerar a compressibilidade do fluido. As principais hipóteses simplificadoras são a de escoamento não viscoso e propriedades constantes em uma determinada seção transversal do bocal. Tais hipóteses representam as equações de Euler quasi-unidmensionais, descritas em sua forma vetorial como

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{1}{S}\frac{\partial (\boldsymbol{F}S)}{\partial x} = \boldsymbol{Q} \tag{1}$$

onde $S = S(x)$ é a área da secção transversal do bocal. O vetor $\boldsymbol{U}$ de variáveis conservadas, o vetor $\boldsymbol{F}$ de fluxos convectivos e o termo fonte $\boldsymbol{Q}$ escritos em termos de quantidades primitividas: densidade (ρ), velocidade (u), energia específica total (e) e pressão (p) são

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho\, v \\ e \end{bmatrix}, \boldsymbol{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (e+p)\, u \end{bmatrix}, \boldsymbol{Q} = \begin{bmatrix} 0 \\ \dfrac{p}{S}\dfrac{\mathrm{dS}}{\mathrm{dx}} \\ 0 \end{bmatrix}$$

a energia total específica correlaciona-se com a energia interna específica (e_i) por

$$e = \rho\left(e_i + \frac{u^2}{2} \right)$$

e o sistema é fechado com a hipótese de gás ideal

$$p = \rho\, R\, T = (\gamma - 1)\, \rho\, e_i$$

O código desenvolvido (`eulerQ1D`) resolve a equação (1) utilizando o método dos volumes fínitos, a discretização espacial segue o método *Steger-Warming* e os fluxos foram avaliados utilizando o método AUSM. A solução em regime estacionário é obtida por meio de marcha no tempo utilizando um integrador *Runge-Kutta* de quarta ordem.

O domínio computacional foi discretizado em 99 elementos finitos, e o critério de convergência das simulações numéricas foi o resíduo $<1e-6$. O objetivo final é otimizar apenas a geometria e portanto todos os casos considerados consideram um bocal operando com Ar ($\gamma = 1.4, R = 287\, J/kg/K$) e nas condições de contorno

$$M_{\mathrm{in}} = 0.01 \qquad p_{0\mathrm{in}} = 104074.0\, Pa \qquad p_e = 0.65 \cdot p_{0\mathrm{in}}$$

Nessas condições, um bocal parametrizado pela equação (2), com $x_t = 5.5$ apresenta a formação de uma onde de choque. Afim de avaliar a capacidade dos métodos de otimização em lidar com discontinuidades, esse foi o bocal escolhido como objetivo. A Figura 1, mostra a geometria que produz a distribuição de pressão alvo $p^\star$.

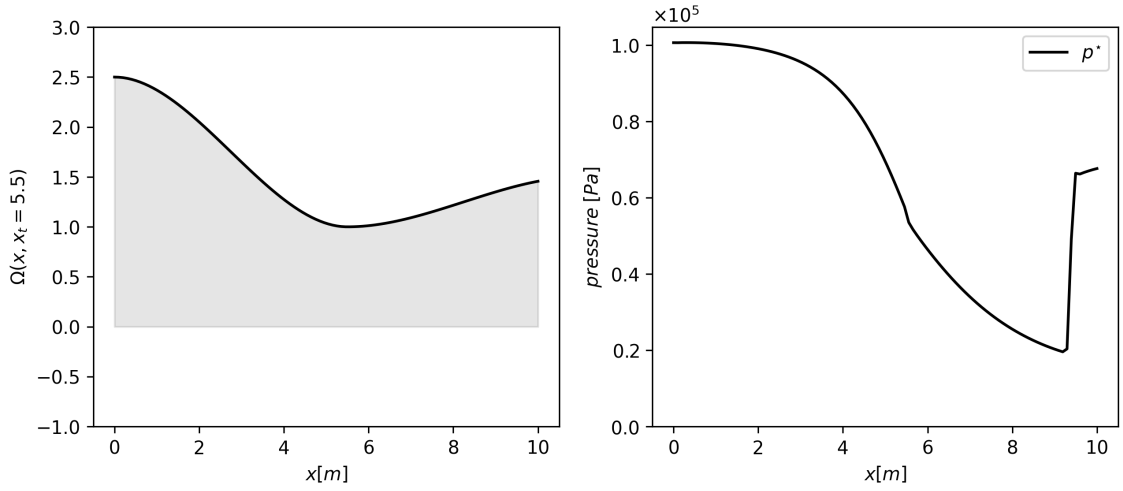

Figura 1. À esqueda a geometria que produz a distribuição de pressão objetivo.

3.2 Método de otimização, `trust-constr`

Diversos algoritmos de otimização foram testados usando duas formulações para o problema inverso, uma dependente de apenas uma variável, e outra multivariável. Embora a taxa de sucesso dos métodos testados tenha sido elevada para o problema com uma única variável, o mesmo não ocorreu para o problema multivariável, nesse último, foram realizados inúmeros experimentos numéricos, que apontaram o método `trust-constr`, como sendo o mais robusto. Em sua for mais geral, com condições de restrição de ingualdade, esse método é uma implementação to algoritmo de ponto interior proposto por [2]. O método pode ser visto como uma combinação de programação linear sequencial quadrática, utilizada para tratar possíveis restrições não lineares e técnicas como ponto interiro e *trust-region* no tratamente de problemas convexos e não convexos. Dado um problema genérico de otimização, dado por

$$\underset{\mathbf{x}}{\text{minimizar}}\ f(\mathbf{x})$$

, sujeito à

$$\begin{aligned} h(\mathbf{x}) &= 0 \\ g(\mathbf{x}) &\leqslant 0 \end{aligned}$$

Onde $h(\mathbf{x})$ representam as restrições de igualdade e $g(\mathbf{x})$ as restrições de inegualdade. O problema é reformulado em termos de subproblemas, resolvíveis por programação linear sequencial quadrática, cada um dos subproblemas assume a forma

$$\underset{\mathbf{x},\mathbf{s}}{\text{minimizar}}\ f(\mathbf{x},\mathbf{s}) - \mu\sum_{i=1}^{m} \ln s_i$$

, sujeito à

$$\begin{aligned} h(\mathbf{x}) &= 0 \\ g(\mathbf{x}) + \mathbf{s} &= 0 \end{aligned}$$

Onde, fazendo μ tender à zero, recupera-se o problema original. Para tratar as restrições de inegualdade, foram introduizidas as variáveis de folga $\mathbf{s}$. Para determinar a solução do problema define-se o Lagrangeano

$$\mathcal{L}(\mathbf{x},\mathbf{s},\boldsymbol{\lambda}_h,\boldsymbol{\lambda}_g) = f(\mathbf{x}) - \mu\sum_{i=1}^{m} \ln s_i + \boldsymbol{\lambda}_h^T h(\mathbf{x}) + \boldsymbol{\lambda}_g^T (g(\mathbf{x}) + \mathbf{s})$$

O subproblema é então solucionado de maneira aproximada, com uma precisão arbitrário ϵ_μ, que é decrementada a cada iteração, de modo que $E(\tilde{\mathbf{x}},\tilde{\mathbf{s}};\mu) \leqslant \epsilon_\mu$, onde

$$E(x,s;\mu) = \max\left(\|\nabla f(\mathbf{x}) + A_h(\mathbf{x})\boldsymbol{\lambda}_h + A_g(\mathbf{x})\boldsymbol{\lambda}_g\|_\infty, \|S\boldsymbol{\lambda}_g - \mu e\|_\infty, \|h(\mathbf{x})\|_\infty, \|g(\mathbf{x}) + \mathbf{s}\|_\infty\right)$$

, onde A_h e A_g são as matrizes de gradientes das restrições, dadas por

$$A_h(\mathbf{x}) = [\nabla h^1(\mathbf{x}), \ldots, \nabla h^t(\mathbf{x})] \qquad A_g(\mathbf{x}) = [\nabla g^1(\mathbf{x}), \ldots, \nabla g^m(\mathbf{x})]$$

Para manter a dimensionalidade correta, $e = [1, \ldots, 1]^T$, e $S = \text{diag}(s^1, \ldots, s^m)$.

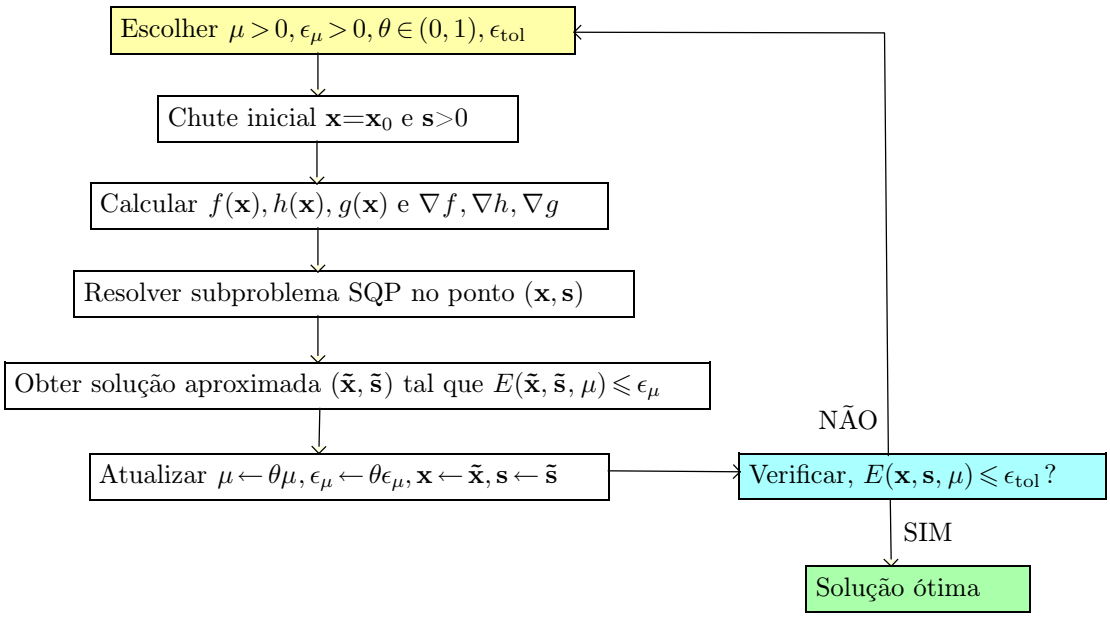

Figura 2. Fluxograma para algoritmo de ponto interior `trust-constr`.

4 Formulação do Problema

O problema inverso em questão pode ser formulado como um problema de otimização, no qual pretende-se minimizar uma função objetivo que quantifica quanto o escomento produzido pela geometria dista de um escoamento previamente estabelecido como alvo. Nesse trabalho, pretende-se obter a geometria Ω de um bocal convergente-divergente que forneca uma determinada distriuição pressão alvo $p^\star$. A métrica de erro utilizada é a média da norma L_2, e a função objetivo J, pode então ser escrita como

$$J(\Omega) = \frac{1}{N}\sqrt{\sum_{i=1}^{N} (p^\star - p)^2}$$

, onde N representa o numéro de elementos utilizados na discretização do modelo numérico. A distribuição de pressão $p = p(\Omega; M_{\text{in}}, p_{0\text{in}}, p_e, \gamma, R)$ é solução das equações de Euler para o escoamento, que por sua vez dependem das condições de contorno especificadas $(M_{\text{in}}, p_{0\text{in}}, p_e, \gamma, R)$, bem como do fluído (γ, R). Essas condições são conhecidas, e portando serão constantes durante todo o processo de otimização, dessa forma a função objetivo é função apenas da geometria do bocal Ω. Nesse estudo foram analisadas duas parametrizações para Ω, uma delas utilizando um polinômio de terceiro grau e noutra uma spline cúbica com 8 pontos.

Foram propostas duas formas de solucionar o problema inverso, utilizando um polinômio com apenas um parâmetro de controle, e uma *spline* cúbica com 8 pontos. Em ambos os casos foram aplicadas restrições nos algoritmos que assim permitem, são eles: `Nelder-Mead`, `L-BFGS-B`, `TNC`, `SLSQP`, `Powell`, e `trust-constr`.

4.1 Parametrização com polinômio

O polinômio proposto em [3], determina uma distribuição de áreas de seção transversal que pode ser ajustada por meio de uma único parâmetro x_t, que controla a coordenada da localiazação da garganta. Para fins ilustrativos, diferentes distribuições de área são mostradas na Figura 3.

$$\Omega(x_t) = \begin{cases} 2.5 + 3\left(\frac{x}{x_t} - 1.5\right)\left(\frac{x}{x_t}\right)^2 & , x < x_t \\ 3.5 - \left(\frac{x}{x_t}\right)\left(6 - 4.5\left(\frac{x}{x_t}\right) + \left(\frac{x}{x_t}\right)^2\right) & , x \geqslant x_t \end{cases} \tag{2}$$

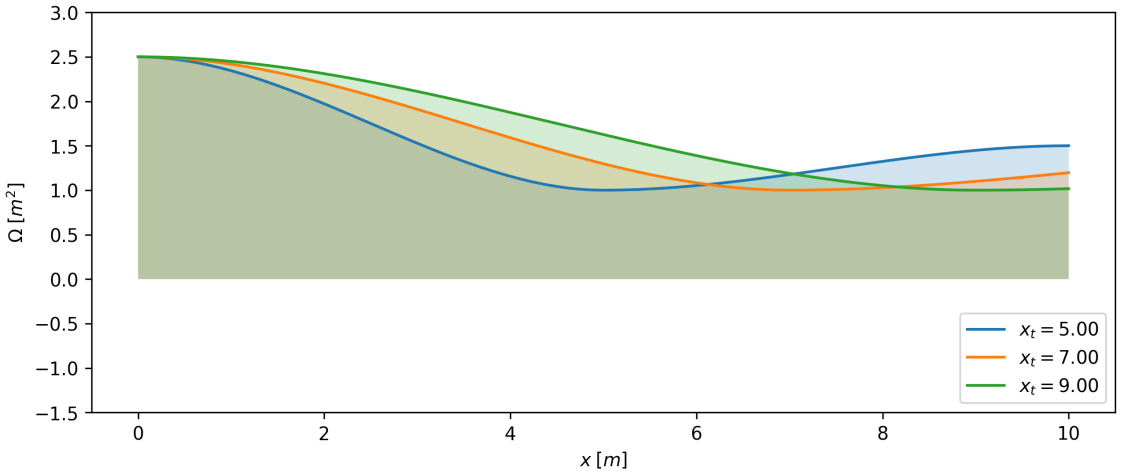

Figura 3. Distribuição de área polinomial, determinada pela equação (2).

Nesse tipo de parametrição o problema inverso pode ser escrito como um problema de otimização que consite em

$$\underset{x_t}{\text{minimizar}}\, J(\Omega(x_t))$$

, sujeito à

$$5 \leqslant x_t \leqslant 10$$

Esse intervalo de restrição foi escolhido para que a distribuição de áreas determinada pela equação (2) corresponda à um bocal do tipo convergente-divergente.

4.2 Parametrição com *Spline* cúbica

A parametrização geométrica utilizando *splines* é bastante adequada aos propósitos de projetos em engenharia uma vez que premite uma representação de curvas e superfícies suaves, com grande capacidade de generalização e utilizando um número finito de pontos de controle discretos.

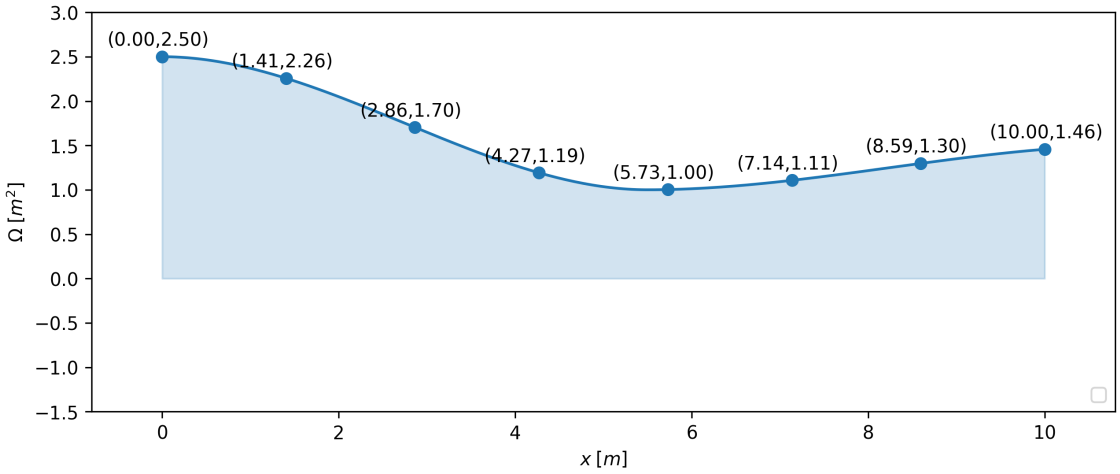

Figura 4. Parametrização utilizando uma *spline* cúbica com 8 pontos de controle.

Nesse problema as coordenadas verticais de cada um dos pontos de controle representa uma variável do problema, a fim a capacidade dos método em resolver o problema da maneira mais generalista possível a restrição imposta foi bastante branda, restringindo todos os pontos de controle ao valor máximo admissível para a área, que é 2.5, o problema de otimização nesse caso é dado por

$$\underset{x_t}{\text{minimizar}}\, J(\Omega(\mathbf{x}))$$

, sujeito à

$$0 < x_i \leqslant 2.5 \qquad i = \{1, 2, \ldots, 8\}$$

5 Implementação Numérica

O métodos de otimização utilizados são parte da biblioteca `scipy`. Os seguintes algoritmos determinísticos foram analisados:

- `Nelder-Mead` - Algoritmo simplex, [4, 5, 6].
- `Powell` - Método de Powell modificado, [7, 8].
- `CG` - Variant of Fletcher-Reeves, [9].
- `BFGS` - Método quasi-Newton de Broyden, Fletcher, Goldfarb, and Shanno, [9].
- `L-BFGS-B` - Modificação do método BFGS para problemas restritos e pouca memória [10].
- `TNC` - Newton truncado, [9, 11].

- `COBYLA` - Método de busca direta, otimazação com restrição por aproximação linear, [12, 13, 14].

- `SLSQP` - Programação linear sequencial, mínimos quadrados, [15].

- `trust-constr` - Algoritmo de região confiável, o `scipy` alterna as implementações, no caso de restrições de igualdade, utiliza *Byrd-Omojokun Trust-Region SQP* [16, 9], para restriçõees de inegualdade muda para *trust-region interior point* [2].

Para o desenvolvimento dessa atividade foi desenvolvido um script em `Python`, as dependências para utilização do script são: `matplotlib, numpy, scipy`, `pandas` e o módulo `solvers`, desenvolvido para executar o solver para o escoamento quasi-unidimensional. O script é rodado em linha de comando, uma descrição de cada parâmetro pode ser acessada com o comando: `python optXti.py -h`. Os códigos necessários para execução estão disponíveis no repositório https://github.com/properallan/MEC326otimizacao.

6 Resultados

6.1 Utilizando parametrização polinomial

A Tabela 1 apresenta os resultados em termos de número de avaliações da função objetivo J e também o valor final ótimo obtido para a função de custo. Nota-se que o valor absoluto para a função objetivo é muito próximo à zero, nesse sentido todos os algoritmos foram capazes de minimizar satisfatóriamente a função objetivo. Quanto ao custo computacional, destaca-se o algoritmo `COBYLA`, apesar de não serem considerados o fator da complexidade de cada algoritmo de otimização, entende-se aqui, que o custo de avaliação da função J é mais relevante, uma vez que envolve a solução de um problema de CFD que pode ser extremamente custoso. Além disso, o algoritmo `COBYLA` não faz uso de derivadas primeiras ou segundas, e portando o número de iterações do algoritmo iguala-se ao número de avaliações da função de custo.

Método	# avaliações	# iterações	J	$x_t^\star$
`SLSQP`	19	7	9.747654620565285e-07	5.49989164
`CG`	129	1	6.008294780406277	6.26775031
`COBYLA`	15	15	8.320969553317746e-05	5.501
`BFGS`	26	7	3.649026309966539e-12	5.50000021
`L-BFGS-B`	32	7	4.841173481412402e-07	5.49992363
`Powell`	75	5	6.347706306802616e-26	5.5
`trust-constr`	42	19	4.139072676691261e-06	5.50022329
`TNC`	30	3	1.961991291866891e-07	5.50004861
`Nelder-Mead`	24	14	4.9622496761630885e-06	5.49975586

Tabela 1. Performance dos métodos de otimização deterministicos para uma parametrização polinomial.

A Figura 5 mostra o gráfico de convergência da função de custo para os métodos analisados. Nesse problema os métodos `COBYLA` e `Powell`, obtiveram as melhores razões de convergência, sendo equivalentes nesse aspecto. O critério de parada utilizadao foi uma diferençã entre à valores para a função de custo $<1e-3$, o método de `Powell`, no entando, seguiu iterando até obter funções de custo com valores na ordem de 6e-27. O método `trust-constr`, apontado na biblioteca `scipy` como

sendo o método mais robusto e confiável, mostrou que tamaha robustez reflete-se em um elevado custo, evidenciado pelo número de iterações e baixa taxa de convergência.

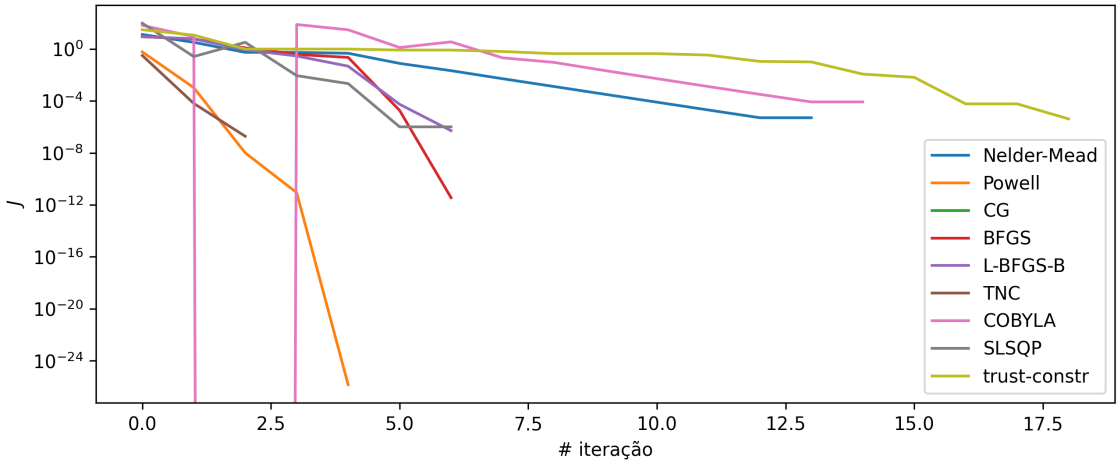

Figura 5. Convergência dos métodos analisados utlizando uma parametrização polinomial com apenas uma variável de projeto.

De maneira geral, exceto o algoritmo `CG`, todo os demais recuperaram muito precisamente a geometria alvo, como mostra a Figura 6.

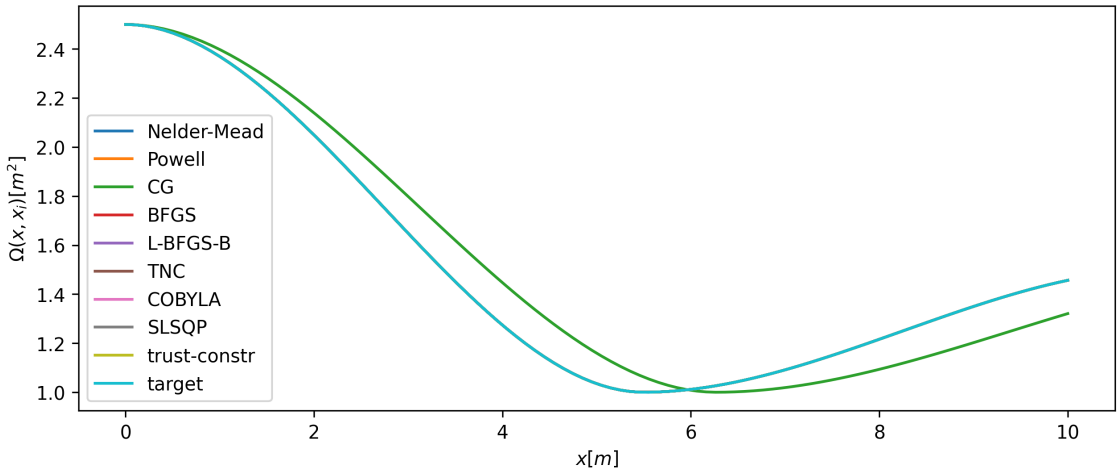

Figura 6. Resultados da otimização para o modelo paramétrico com uma única variável.

6.2 Utilizando parametrização com *spline* cúbica

A Tabela 2 apresenta a performance dos métodos testados utilizando uma parametrização de *spline* cúbinca. Nesse caso, o número de variáveis aumentou de 1 para 8, e o problema tornou-se excessivamente custoso. Nenhum dos métodos em questão foi capaz de recuperar a geometria alvo. O fator de tolerância utilizado foi tol $< 1 - e3$, acredita-se que uma tolerância menor, resultaria em um número maior de iterações com resultados potencialmente melhores, observa-se no entanto que o problema inverso é mal posto, uma vez que não há garantia de existência e unicidade da sulução numérica.

Método	# avaliações	# iterações	J
`SLSQP`	233	22	0.02003277456604602
`CG`	7715	389	0.011622164354443488
`COBYLA`	422	422	0.24756052150364757
`BFGS`	1194	69	0.008818300170018417
`L-BFGS-B`	279	24	0.1267727940841186
`Powell`	2635	26	0.009481766589186796
`trust-constr`	1620	170	0.011477187238382045
`TNC`	909	15	0.4344095665774871
`Nelder-Mead`	580	380	0.14700252215240825

Tabela 2. Performance dos método para uma parametrização com *spline* cúbica.

A Figura 7, mostra a geometria final recuperada pelo método `trust-constr`, nesse caso os pontos de controle são apenas os interiores. Mesmo com um elavado número de iterações, não foi possível recuperar de forma satisfatória a geometria do bocal, sobretudo na região de entrada.

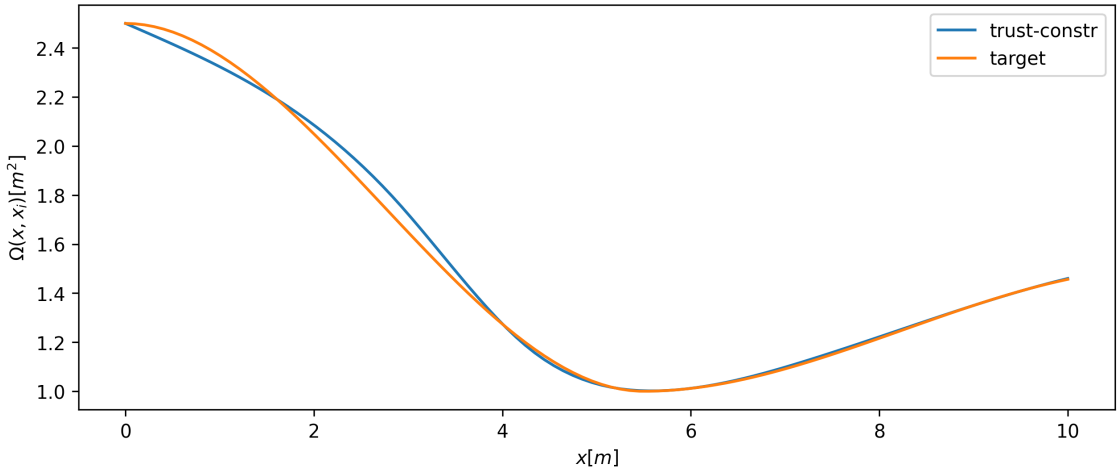

Figura 7. Resultado do projeto inverso utilizando o método `trust-constr` e restrições de igualdade para as áreas na entrada e na saída do bocal. O processo tomou assombrosos 8722 avaliações da função objetivo.

6.2.1 Restrições mais rígidas

A fim de tornar as dificuldades relacionadas ao problema inverso mal posto, foram aplicadas condições de restrições mais rígidas, fixando-se as àreas na entrada e na saída do bocal para os valores alvo, nesse caso o problema de otimização torna-se

$$\underset{x_t}{\text{minimizar}}\, J(\Omega(x_t))$$

, sujeito à

$$\begin{aligned} x_1 &= 2.5 \\ 0 < x_i &\leqslant 2.5 \qquad i = \{2, \ldots, 7\} \\ x_8 &= 1.5 \end{aligned}$$

7 Discussões

A ferramenta `scipy` disponibiliza uma série de bons algoritmos de otimização prontos para serem utilizados de maneira bastante intuitíva. Escolher qual otimizador é mais apropriado depende muito do problema em questão, e pode tornar-se uma tarefa bastante trabalhosa em casos de otmização multivariável, sobretudo de o problema é mal posto, como no caso do projeto inverso do bocal convergente-divergente, onde nem a unicidade, nem a existência das soluções é garantido.

Embora disponíveis, não foram testados os algoritmos estocásticos, que poderiam lidar melhor com esse tipo de problema, onde a avaliação de derivadas pode ser problemática e custusa computacionalmente.

Nota-se que na formulação do problema de otimização, as restrições são escolhas muito relevantes, podendo, caso mal estabelecidas, inviabilizar o processo de otimização.

8 Conclusão

Diversos algoritmos de otimização foram analisados na solução do problema proposto, no caso do projeto inverso com apenas uma variável, exceto o algoritmo `CG`, todos os demais apresentaram um excelente desempenho, recuperando de maneira muito precisa a geometria objetivo. No caso da otimização multivariável o problema mostrou-se bastante desafiador, uma vez que os algoritmos convergem para pontos de mínimo que não são representam à geometria objetivo, tal comportamento deve-se ao problema inverso em questão ser mal posto por natureza. Para aliviar esse problema restrições mais severas foram impostas, e desse modo o algortimo `trust-constr`, que fora brevemente descrito, pode recuperar a geometria alvo, ainda que, de maneira menos eficiente que a parametrização polinomial, como esperado, o custo da otimização multivariável envolvendo solução de CFD cresce excessivamente comparado ao caso uni-variável.

Bibliografia

[1] Markus Rumpfkeil e David Zingg. A general framework for the optimal control of unsteady flows with applications. In *45th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, jan 2007.

[2] Richard H. Byrd, Mary E. Hribar, e Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, jan 1999.

[3] Geum-Su Yeom e Jung-Il Choi. Efficient exact solution procedure for quasi-one-dimensional nozzle flows with stiffened-gas equation of state. *International Journal of Heat and Mass Transfer*, 137:523–533, jul 2019.

[4] Fuchang Gao e Lixing Han. Implementing the nelder-mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277, may 2010.

[5] J. A. Nelder e R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, jan 1965.

[6] Margaret H. Wright. Direct search methods: once scorned, now respectable. 1996.

[7] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, feb 1964.

[8] William H. Press, Saul A. Teukolsky, William T. Vetterling, e Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007.

[9] J. Nocedal e S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[10] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, e Jorge Nocedal. Algorithm 778: l-BFGS-B. *ACM Transactions on Mathematical Software*, 23(4):550–560, dec 1997.

[11] Stephen G. Nash. Newton-type minimization via the lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–788, aug 1984.

[12] M. J. D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336, jan 1998.

[13] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. 1994.

[14] M. J. D. Powell. A view of algorithms for optimization without derivatives 1. 2007.

[15] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.

[16] Marucha Lalee, Jorge Nocedal, e Todd Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM Journal on Optimization*, 8(3):682–706, aug 1998.