Trabalho III
Otimização - 2015.2
Universidade Federal do Rio de Janeiro

Felipe Matheus Fernandes Oliveira
felipemfo@poli.ufrj.br

Henrique Mattos
henriquemattos@poli.ufrj.br

Jean Américo Tomé
jamerico@poli.ufrj.br

Philipe Miranda de Moura
philipemoura@poli.ufrj.br

17 de Março de 2016

Conteúdo

1 Introdução

Durante toda a história da ciência, sempre foi necessário o desenvolvimento de métodos analíticos para interpretar sistemas relacionados a área da física, matemática e engenharia. Entretanto, quando passamos para o plano empírico muitos erros ocorrem e desviam os valores reais dos valores esperados, erros esses podem ser classificados como sistemáticos e estatísticos [1], associados respectivamente a instrumentos de medição e fatores aleatórios.

Com isso, muitas vezes quando desejamos estabelecer uma função baseada em medidas tomadas, aparecem erros gerando inconsistências caso analisemos literalmente os métodos analíticos. Uma maneira muito efetiva de contornar esse problema é o método dos **mínimos quadrados**, que consiste na análise do quadrado da diferença entre o valor obtido da medida e o valor analítico para determinado ponto.

Nesse trabalho aplicaremos esse conceito em um problema de controle no qual deseja se achar os parâmetros ω_n e ζ de um SLIT através da medida da resposta ao degrau unitário em 10 pontos.

Após aplicarmos o método dos mínimos quadrados, devido ao modelo analítico de equação que possuímos, teremos duas variáveis da função erro cujo valor desejamos minimizar. Com esse objetivo aplicaremos dois métodos de otimização multivariável para acharmos valores ótimos das duas variáveis em questão, podendo assim modelar nosso SLIT com maior perfeição. Os métodos utilizados serão métodos de otimização da classe dos "métodos pobres", pois não possuem alto nível matemático, como cálculo de gradientes vetoriais e outros algebrismos, mas se baseiam em algoritmos de tentativa e erro. Essas alternativas são necessárias devido a não-linearidade da equação analítica (como é o caso aqui tratado). Esses métodos serão **Hooke & Jeeves** e o **Algoritmo genético**, que terão seus algoritmos explicados ao longo do trabalho.

2 Apresentação e discussão do Problema

Primeiramente nos foram dadas as medidas da resposta do sistema ao degrau em cada ponto. Elas seguem listadas na tabela abaixo.

Tabela 1: Resposta ao degrau do SLIT.

m(t)	t
0.25	1.00
0.56	2.00
0.72	3.00
0.81	4.00
0.89	5.00
0.92	6.00
0.97	7.00
0.97	8.00
0.98	9.00
0.99	10.00

Sabemos pelo modelo analítico que determinado sistema é caracterizado pela seguinte equação:

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{1}$$

E sua resposta no domínio do tempo ao degrau unitário é:

$$h(t) = 1 + \frac{p_2 e^{p_1 t} - p_1 e^{p_2 t}}{p_1 - p_2} \tag{2}$$

Onde p_1 e p_2 são pólos da equação, relacionados com ζ e ω_n pela resolução da equação de Bhaskara:

$$p_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1} \tag{3}$$

O primeiro passo para conseguirmos modelar esse problema é calcular a função a ser minimizada, que pela fórmula dos mínimos quadrados, obtemos:

$$\Delta = \sum_{i=1}^{n} (h(t) - m(t))^2 \tag{4}$$

Onde Δ é a função erro a ser minimizada, e em cada parcela do somatório, $m(t_i)$ é o valor empírico do ponto t_i e $y(t_i)$ é o valor analítico da função, sendo n igual ao total de pontos amostrados.

Após essa etapa, teremos uma função que consiste de um somatório de n parcelas sendo que cada uma delas está em função de duas variáveis p_1 e p_2 .

$$\Delta(p_1, p_2) = \sum_{i=1}^{n} [(1 + \frac{p_2 e^{p_1 t_i} - p_1 e^{p_2 t_i}}{p_1 - p_2}) - m(t_i)]^2 \tag{5}$$

Devido a não-linearidade da função, aplicaremos os métodos numéricos em questão a fim de obter os valores otimizados de p_1 e p_2.

Para alcançar o objetivo do trabalho, uma vez que p_1 e p_2 já foram estabelecidos pelos métodos em questão, substituiremos nas duas equações que a fórmula de Bhaskara nos oferece, achando assim um sistema de solução única que nos fornecerá os valores ótimos de ω_n e ζ.

$$p_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1} \tag{6}$$

$$p_2 = -\zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1} \tag{7}$$

3 Hooke & Jeeves

O método de Hooke & Jeeves [2] é um método de otimização numérica pertencente a família de pesquisa padrão. Nela busca se resolver problemas sem precisar do Gradiente da função a ser otimizada, sendo muito útil em funções não contínuas e não diferenciáveis.

Aqui trataremos de explicar tanto o método quanto sua implementação no MATLAB, a fim de ajudar no entendimento do desenvolvimento do programa e futuramente à resolução do problema proposto.

Existem algumas formas diferentes de executar o método de Hooke & Jeeves, entretanto todos se baseiam no mesmo princípio da verificação dos arredores do ponto respeitando um incremento (γ) e estabelecendo uma direção de avanço para um novo ponto com valor melhor que o anterior.

Para a explicação do método utilizado, listaremos os tópicos com os passos do algoritmo:

- (i) Verificação e definição do sentido de crescimento de cada variável;
- (ii) Avanço respeitando a direção até ocorrer uma condição de parada;

I - Verificação e definição do sentido de crescimento de cada variável

Se baseia em encontrar o sentido de melhoria da função para cada variável, testando e comparando os valores da função em $f(x)$, $f(x + \gamma)$ e $f(x - \gamma)$. Escolhemos dos três aquele que possui o menor valor (minimização) valor, e estabelecemos que o sentido de otimização para aquela variável será aquele que fora aplicado em x e retornou uma resposta melhor que a original (mais o incremento, menos o incremento, ou sem adição de incremento) . Realizamos esse processo para todas as variáveis, definindo o passo (se devemos adicionar, subtrair o incremento de cada uma, ou apenas deixá la igual) que cada uma deverá tomar na etapa do avanço.

O programa implementado em MATLAB encontra-se no apêndice A.1.

II - Avanço respeitando a direção até ocorrer uma condição de parada

Após o estabelecimento de cada passo para cada variável, realizaremos os cálculos utilizando $2 * \gamma$ para obter os valores da função nos pontos de avanço, e caso obtenha se sucesso continuamos avançando até ocorrer alguma condição de parada.

Existem 3 possíveis condições de parada.

A primeira delas acontece caso o passo I tenha encontrado apenas valores maiores para a função ao realizar a adição e subtração dos incrementos. Em outras palavras, a base atual é melhor do que os pontos verificados. Caso isso ocorra dividimos o incremento por 2 e voltamos ao passo 1 (refinamos a busca).

A segunda acontece caso o valor da função em um dos passos seja maior do que a do passo anterior. Se isso ocorrer devemos voltar ao passo 1 e recalcular o sentido de avanço.

Por fim a última condição de parada é caso o incremento seja menor do que a tolerância estabelecida, acarretando o sucesso do método.

O programa implementado em MATLAB encontra-se no apêndice A.2.

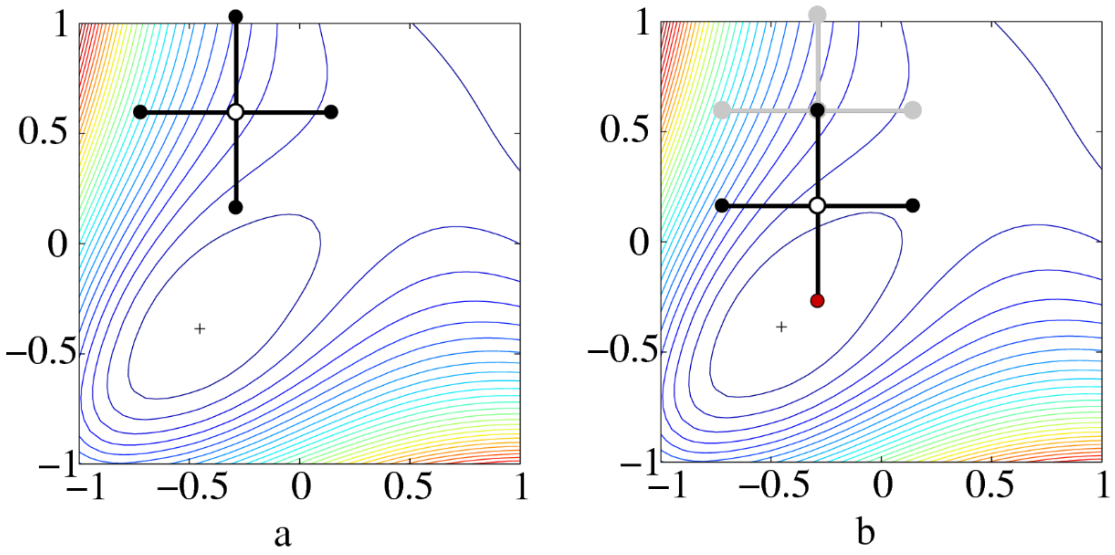

Figura 1: Ilustração do método, passos a b.

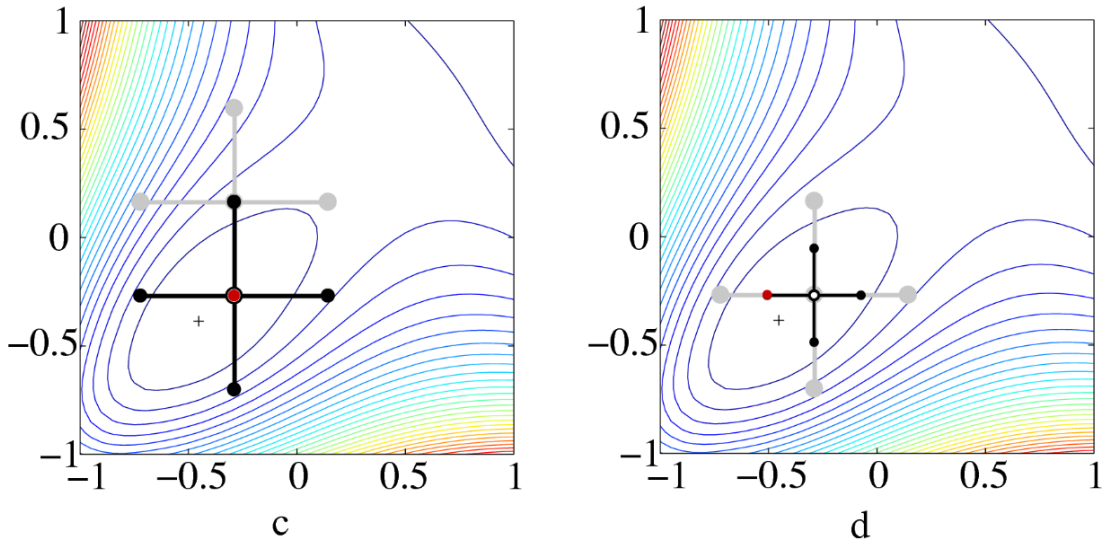

Figura 2: Ilustração do método, passos c d.

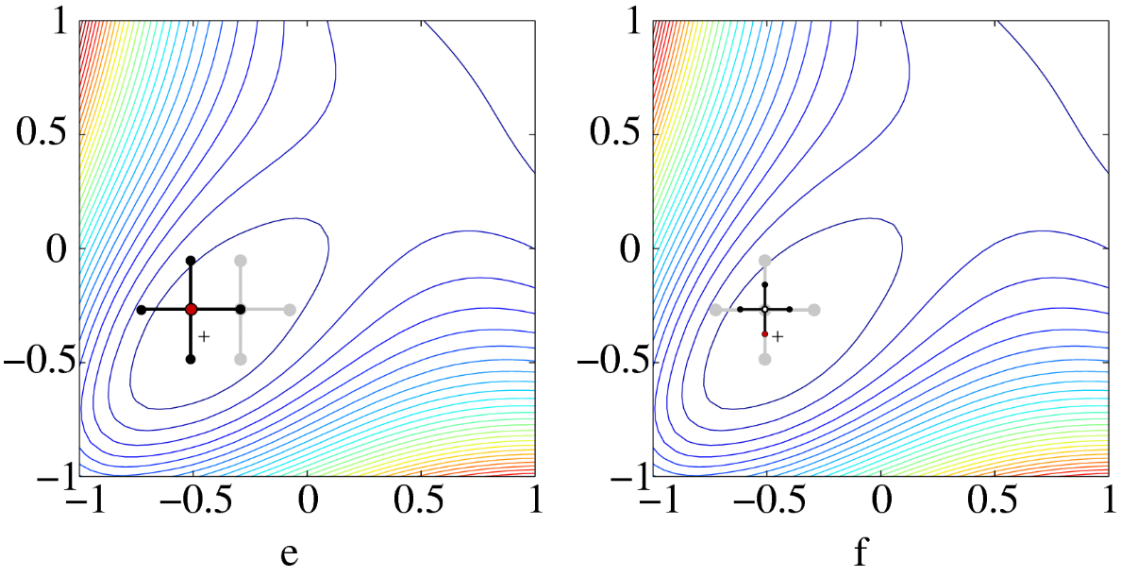

Figura 3: Ilustração do método, passos e f.

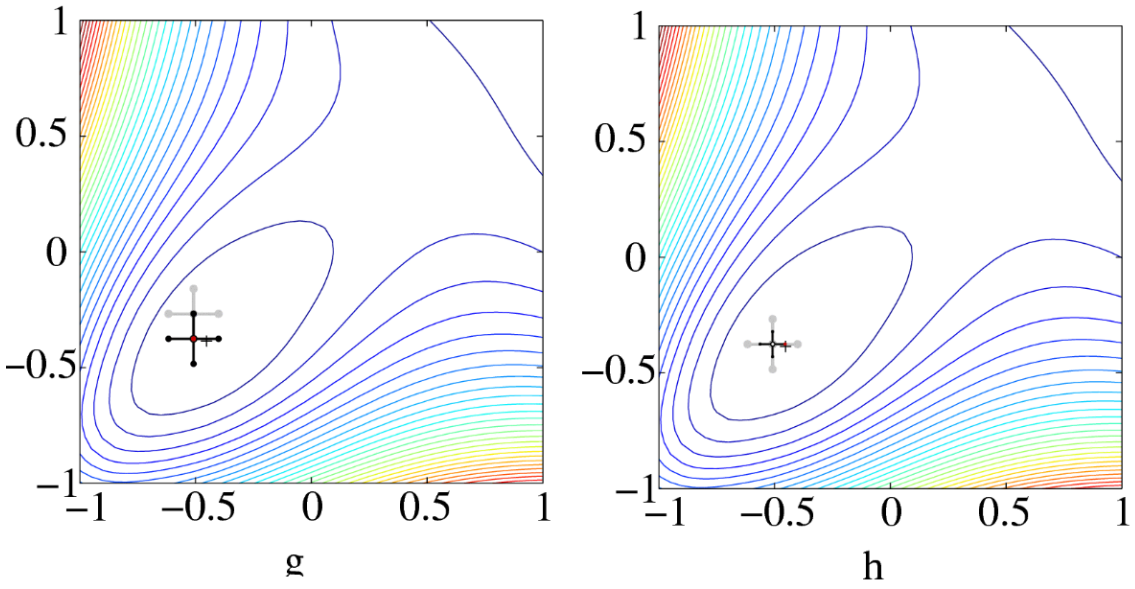

Figura 4: Ilustração do método, passos g h.

3.1 Algoritmo para Hook & Jeeves

Na figura 5, temos um algoritmo esquemático para o método de Hook & Jeeves.

Algoritmo estruturado:
Estaelecer um incremento e uma tolerância para cada variável.
Escolher uma base.
Repetir:
Explorar a vizinhança da base (em busca da direção provável do ótimo)
Se houve sucesso em alguma direção:
Então: progredir (na direção rpovável) até haver um insucesso.
Se não (proximidade do ótimo):
Se chegou ao ótimo:
Então: finalizar
Se não:
reduzir os incrementos.

Figura 5: Algoritmo esquemático para o método Hook & Jeeves

4 Algoritmo Genético

Os métodos de minimização através de algoritmos genéticos compõem uma família de, pelo menos, uma dezena de submétodos. Todos têm o mesmo princípio de funcionamento: a teoria de Darwin de evolução das espécies.

De forma geral, uma população (um conjunto de pontos) é heterogênea se possui indivíduos com diferentes propensões a sobreviver. Isso é, há indivíduos mais fortes que outros. No caso de minimização, um ponto mais forte é aquele cujo valor da função é menor.

Além da seleção de sobreviventes, a teoria da evolução conta também com procriação e mutação de indivíduos.

Neste trabalhos, dividiremos o algoritmo genético em três etapas: geração e seleção, recombinação (crossover) e mutação.

4.1 I - Geração e seleção de indivíduos

Neste trabalho, adotou-se a seguinte geração de indivíduos:
Partindo-se de um vetor inicial $[x, y, z, ..., k]$, gera-se n pontos aleatórios à partir de uma distribuição normal com média centrada na k-ésima dimensão do vetor e com uma variância de ψ, onde este último pode ser ajustando de forma a obter pontos mais ou menos dispersos. Assim, para cada dimensão do vetor geramos m pontos aleatórios. Por fim, através da apêndice desses pontos, teremos outros m vetores. Em nossos testes, a combinação $m = 100$ e $\psi = 5$ demonstrou-se um ótima.

No apêndice A.3, encontramos esse método implementado.

Para a seleção, passamos esses conjunto pontos,população, através da função objetivo e reordenamos o conjunto em ordem crescente. A partir disso, selecionamos os p melhores pontos - isto é, os que geraram os menores valores da função objetivo - e descartamos os demais. Essa é a nossa seleção natural, e os pontos sobreviventes são chamados de pais. Em nossos testes, adotamos $p = 5$.

Nos apêndices A.3, A.4 e A.5 encontramos esses métodos implementados.

4.2 II - Recombinação

Aqui, pegamos os p pais e fazemos uma combinação dois a dois entre eles utilizando um fator de exploração r >0 e um número aleatório r variando entre zero e um. Para cada combinação, geramos outros 2 filhos.

Na figura 6, vemos essa lei de formação, obtida através de [3].

2.1.3.2 Blending (Intermediate) crossover

The mathematic description of this crossover is:

$C_1 = \gamma.P_1 + (1-\gamma).P_2$ (2-2)

$C_2 = (1-\gamma).P_1 + \gamma.P_2$

$\gamma = (1+2.\alpha).r - \alpha$ (2-3)

P_1, P_2 – chromosomes of the parents;
C_1, C_2 – chromosomes of the children (offspring individuals);
α - exploration coefficient – user defined ($\alpha \geq 0$);
r – random number between 0 and 1;

The graphical representation is shown on Figure 2.3 and Figure 2.4.

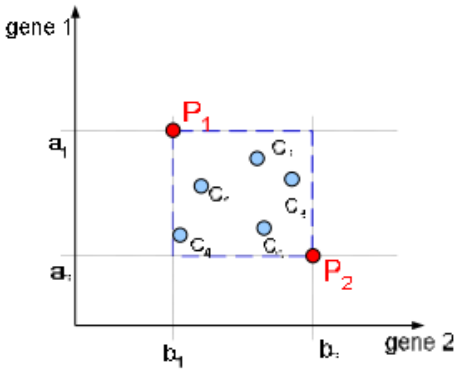

Figure 2.3 – Graphical representation of blending crossover with $\alpha = 0$

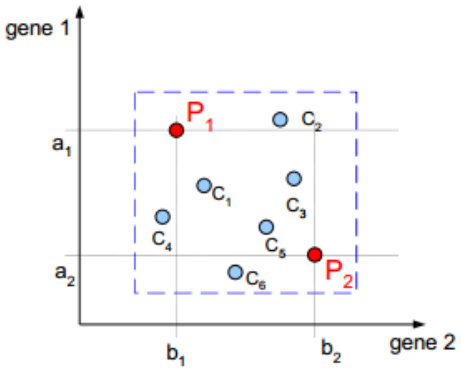

Figure 2.4 – Graphical representation of blending crossover with $\alpha > 0$

Figura 6: Pag. 7 da referência adotada demonstrando a lei de recombinação.

O interessante dessa lei é que, para $r = 0$, limitamos os novos pontos, filhos, a estarem dentro da uma área limitada pelos pais. Mas, quando fazemos $r>0$, podemos explorar as bordas e pontos exteriores, tal como mostra as figuras da referência.

No apêndice A.6, listamos o método implementado.

4.3 III - Mutação

Por fim, realizamos a mutação dos pais e dos filhos gerados. Para este fim, para cada dimensão desses pontos gerados, $[x, y, z, ..., k]$, pegamos a diferença entre o maior e o menor desses valores, definindo um Δ para essa dimensão. E então, para cada dimensão desses pontos gerados, perturbamos ela por um fator de $\gamma \times \Delta$, sendo esse γ um número aleatório resultado de uma distribuição uniforme variando entre $-\beta$ e β, em que este último é um parâmetro definido entre 0 e 1.

Em termos matemáticos, temos, para a k-ésima dimensão dessa população:

$$K2 = K + \gamma * \Delta_K \tag{8}$$

Onde: γ: resultado de uma distribuição aleatória uniforme variando entre $-\beta$ e β, com β entre 0 e 1.

No apêndice A.7, encontramos o código implementado.

Por fim, retornamos à seleção natural e tornamos um loop.

4.4 Algoritmo Genetico

Na figura 7, temos um algoritmo esquemático para o método Genético.

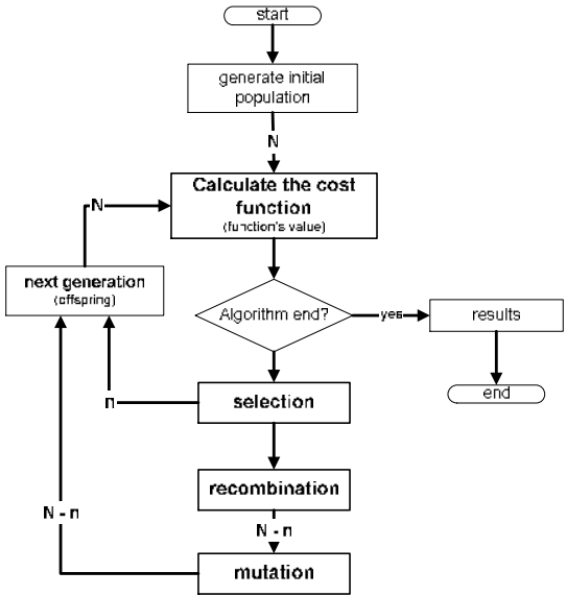

Figure 2.7 – General scheme of the evolutionary algorithms

Figura 7: Algoritmo esquemático para o método Genetico

5 Solução por Hooke & Jeeves

Aqui exporemos os resultados da otimização da equação 5 pelo método de Hooke & Jeeves quando n assume os valores de 3, 5 e 10.

Entretanto, antes de tudo, a fim de ilustrar melhor, encontra se abaixo todos os pontos representados no gráfico de $m(t) \times t$.

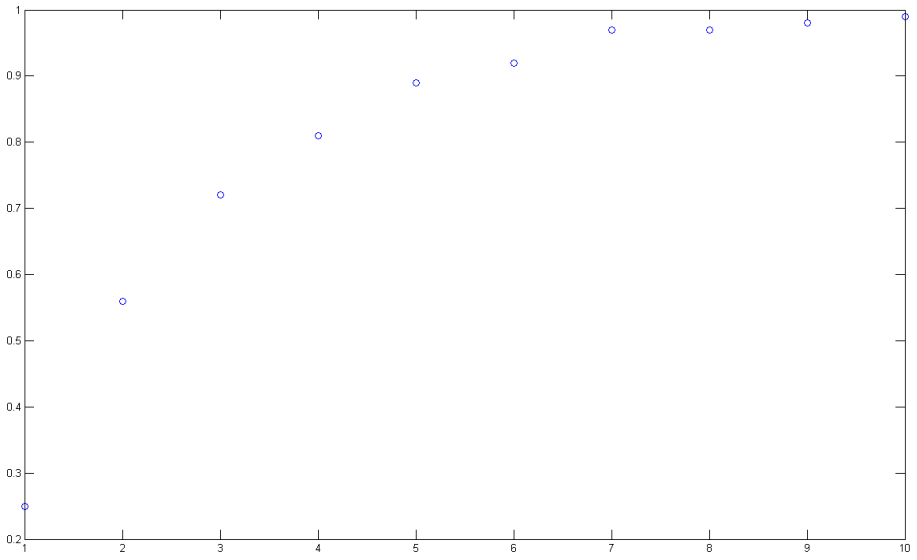

Figura 8: Gráfico $m(t) \times t$

5.1 Para 3 amostras

Pegando como pontos de amostragem os valores nos instantes 1, 3 e 10, obtemos a equação no formato:

```
/ p1 exp(3 p2) - p2 exp(3 p1)    7 \2   / p1 exp(10 p2) - p2 exp(10 p1)    1  \2   / p1 exp(p2) - p2 exp(p1)   3 \2
| -------------------------- - -- |  + | ---------------------------- - --- |  + | ---------------------- - - |
\          p1 - p2             25 /    \            p1 - p2              100 /    \         p1 - p2          4 /
```

Figura 9: Função representada no modo *pretty* do MATLAB

O resultado do programa em sua interface desenvolvida foi:

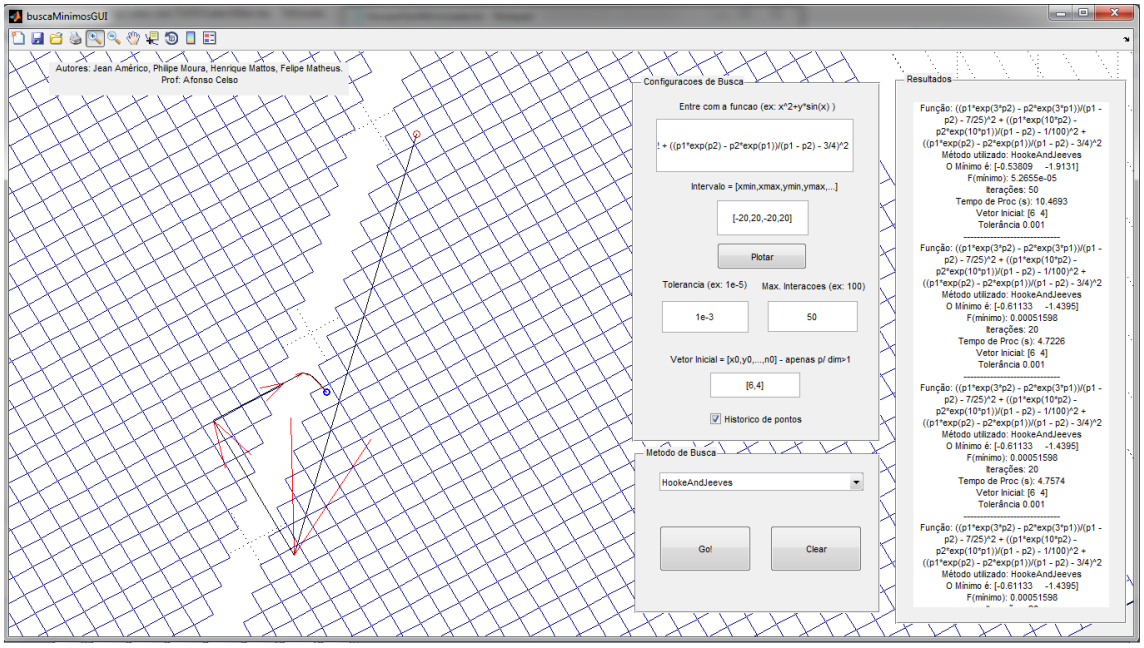

Figura 10: Interface do programa após a otimização achando $p_1 = -0.53809$ e $p_2 = -1.9131$

5.2 Para 5 amostras

Pegando como pontos de amostragem os valores nos instantes 1, 2, 3, 5 e 9, obtemos a equação no formato:

```
/ p1 exp(3 p2) - p2 exp(3 p1)    7 \2   / p1 exp(2 p2) - p2 exp(2 p1)   11 \2   / p1 exp(9 p2) - p2 exp(9 p1)    1 \
\2
| -------------------------- - -- |  + | -------------------------- - -- |  + | -------------------------- - -- |
\          p1 - p2             25 /    \          p1 - p2             25 /    \          p1 - p2             50 /

    / p1 exp(5 p2) - p2 exp(5 p1)    11 \2   / p1 exp(p2) - p2 exp(p1)   3 \2
  + | -------------------------- - --- |  + | ----------------------- - - |
    \          p1 - p2             100 /    \         p1 - p2           4 /
```

Figura 11: Função representada no modo *pretty* do MATLAB

O resultado do programa em sua interface desenvolvida foi:

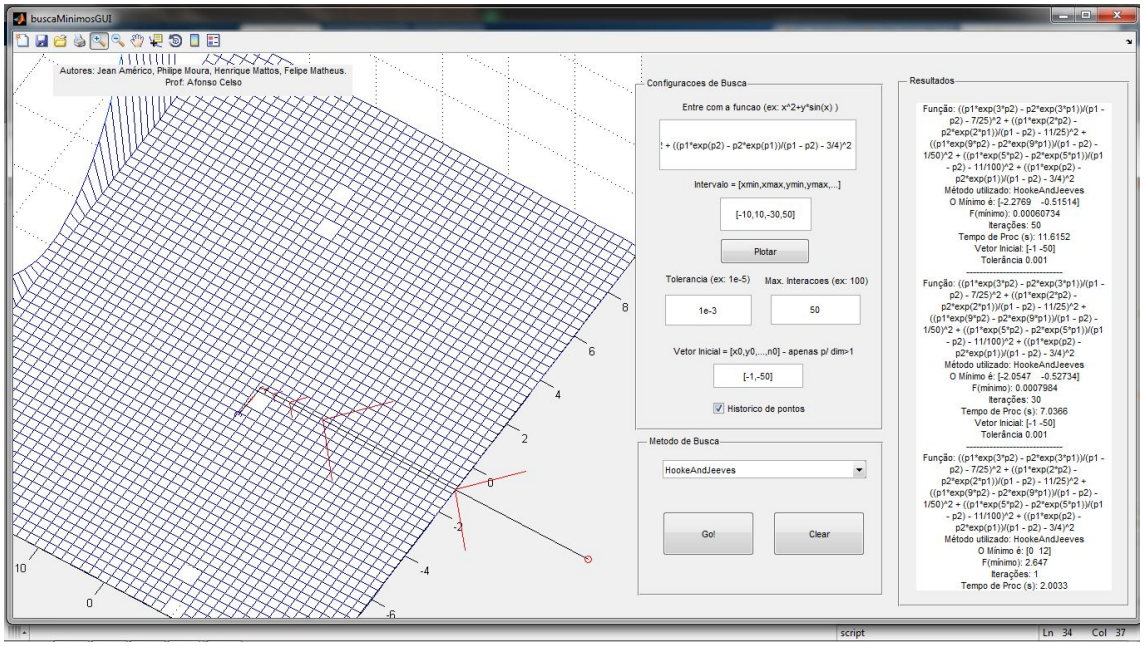

Figura 12: Interface do programa após a otimização achando $p_2 = -2.2769$ e $p_1 = -0.51514$

5.3 Para 10 amostras

Pegando como pontos de amostragem todos os valores, obtemos a equação no formato:

```
/ p1 exp(3 p2) - p2 exp(3 p1)    7 \2   / p1 exp(6 p2) - p2 exp(6 p1)    2 \2   / p1 exp(2 p2) - p2 exp(2 p1)   11 \
\2
| ---------------------------- - -- |  + | ---------------------------- - -- |  + | ---------------------------- - -- |
\           p1 - p2             25 /    \           p1 - p2             25 /    \           p1 - p2             25 /

    / p1 exp(9 p2) - p2 exp(9 p1)    1 \2   / p1 exp(7 p2) - p2 exp(7 p1)    3  \2
  + | ---------------------------- - -- |  + | ---------------------------- - --- |
    \           p1 - p2             50 /    \           p1 - p2             100 /

    / p1 exp(8 p2) - p2 exp(8 p1)    3  \2   / p1 exp(5 p2) - p2 exp(5 p1)    11 \2
  + | ---------------------------- - --- |  + | ---------------------------- - --- |
    \           p1 - p2             100 /    \           p1 - p2             100 /

    / p1 exp(10 p2) - p2 exp(10 p1)    1  \2   / p1 exp(4 p2) - p2 exp(4 p1)    19 \2
  + | ----------------------------- - --- |  + | ---------------------------- - --- |
    \            p1 - p2              100 /    \           p1 - p2             100 /

    / p1 exp(p2) - p2 exp(p1)   3 \2
  + | ---------------------- - - |
    \         p1 - p2           4 /
```

Figura 13: Função representada no modo *pretty* do MATLAB

O resultado do programa em sua interface desenvolvida foi:

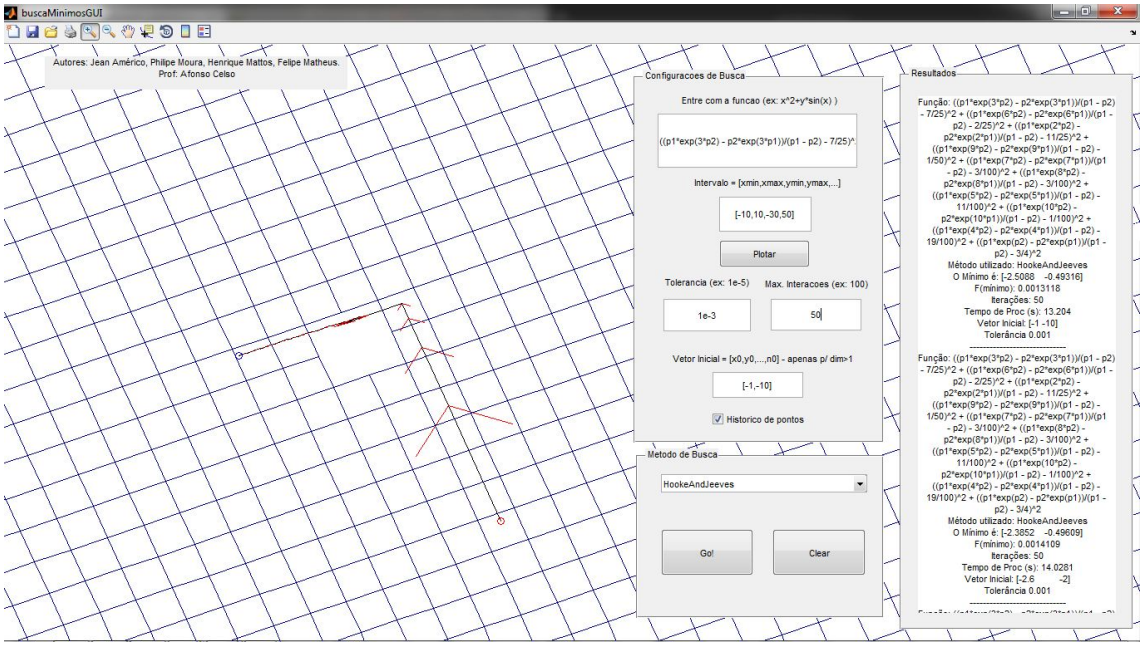

Figura 14: Interface do programa após a otimização achando $p_2 = -2.5088$ e $p_1 = -0.49316$

5.4 Resultado final

Por fim, tendo em vista que obtemos os valores de p_1 e p_2 para cada um dos 3 casos, o que nos fez chegar a 3 diferentes equações (pela substituição na equação 2), plotaremos cada equação substituindo t pelos valores de 1 até 10.

Assim, através da análise abaixo, percebemos que quanto maior o número de pontos de amostragem utilizados no método de mínimos quadrados, o gráfico final se enquadrará melhor aos pontos.

Para concluir, utilizamos os valores de p_1 e p_2 para o caso de 10 amostras nas equações (6) e (7), obtendo os valores de $\zeta = 1.34942$ e $\omega_n = 1.11231$.

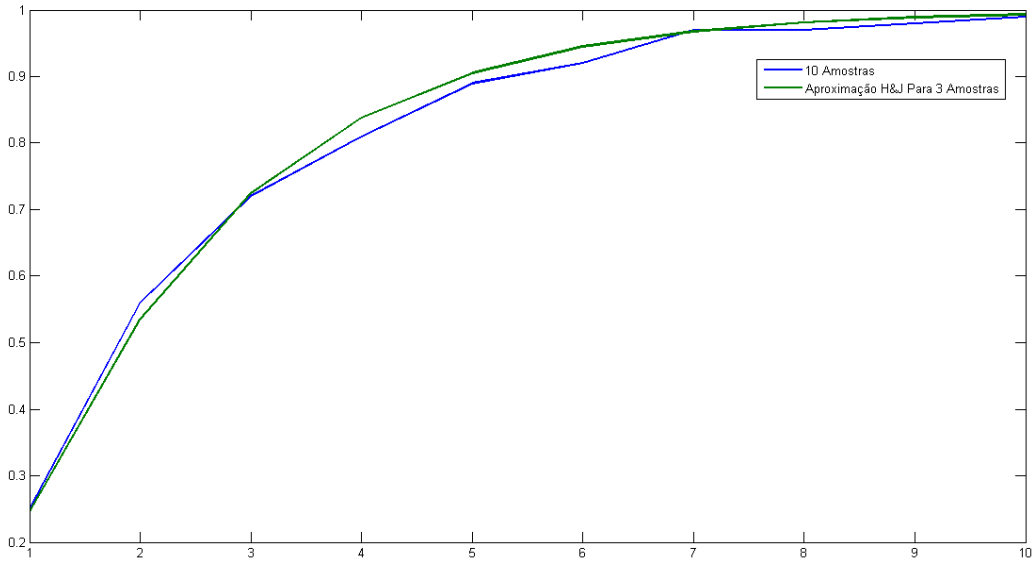

Figura 15: Função com os valores de p_1 e p_2 obtidos no 5.1

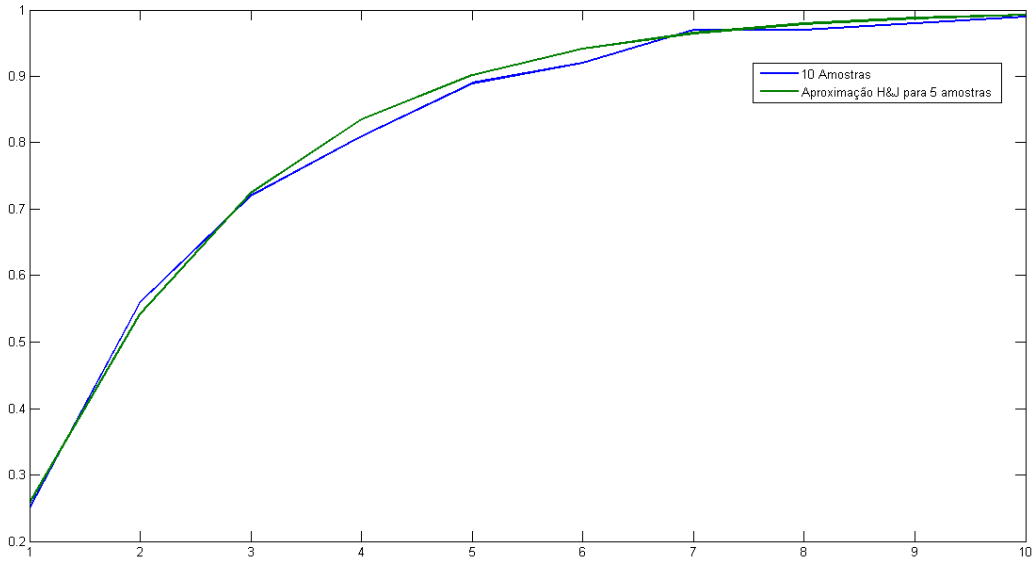

Figura 16: Função com os valores de p_1 e p_2 obtidos no 5.2

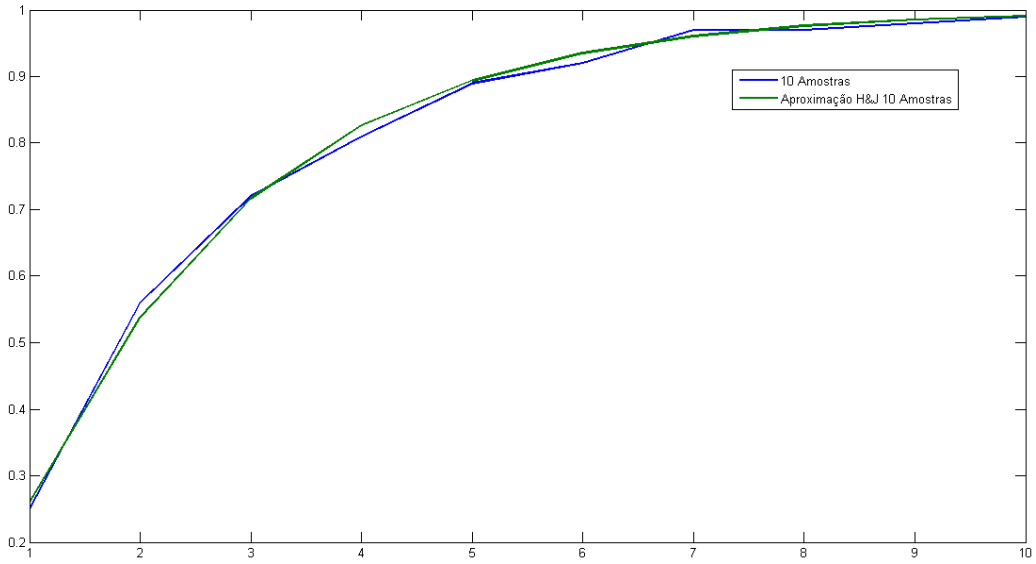

Figura 17: Função com os valores de p_1 e p_2 obtidos no 5.3

6 Solução pelo algoritmo genético

Aqui exporemos os resultados da otimização da equação 5 pelo método do algoritmo genético quando n assume os valores de 3, 5 e 10.

6.1 Para 3 amostras

Pegando como pontos de amostragem os valores nos instantes 1, 3 e 10, obtemos a equação no formato:

```
/ p1 exp(3 p2) - p2 exp(3 p1)    7 \2   / p1 exp(10 p2) - p2 exp(10 p1)    1  \2   / p1 exp(p2) - p2 exp(p1)   3 \2
| ------------------------- - -- |  + | --------------------------- - --- |  + | ---------------------- - - |
\          p1 - p2            25 /    \           p1 - p2             100 /    \         p1 - p2          4 /
```

Figura 18: Função representada no modo *pretty* do MATLAB

O resultado do programa em sua interface desenvolvida foi:

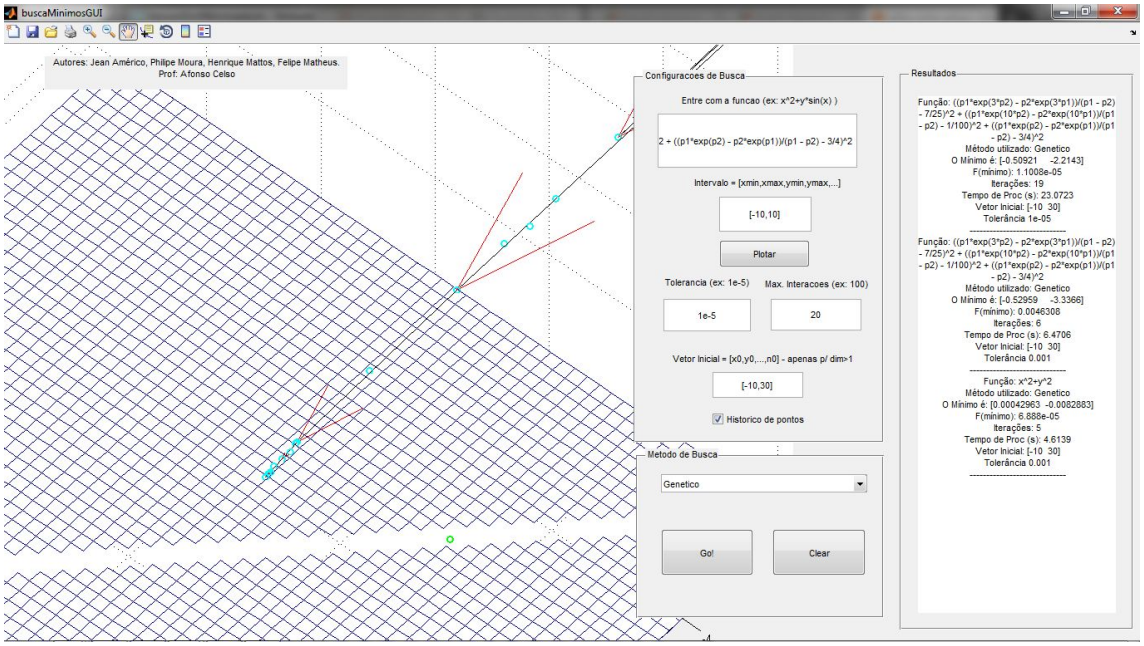

Figura 19: Interface do programa após a otimização achando $p_1 = -0.50921$ e $p_2 = -2.2143$

6.2 Para 5 amostras

Pegando como pontos de amostragem os valores nos instantes 1, 2, 3, 5 e 9, obtemos a equação no formato:

```
/ p1 exp(3 p2) - p2 exp(3 p1)    7 \2   / p1 exp(2 p2) - p2 exp(2 p1)   11 \2   / p1 exp(9 p2) - p2 exp(9 p1)    1 \
\2
| -------------------------- - -- |  + | -------------------------- - -- |  + | -------------------------- - -- |
\          p1 - p2             25 /    \          p1 - p2             25 /    \          p1 - p2             50 /

    / p1 exp(5 p2) - p2 exp(5 p1)    11 \2   / p1 exp(p2) - p2 exp(p1)   3 \2
  + | -------------------------- - --- |  + | ----------------------- - - |
    \          p1 - p2            100 /    \         p1 - p2           4 /
```

Figura 20: Função representada no modo *pretty* do MATLAB

O resultado do programa em sua interface desenvolvida foi:

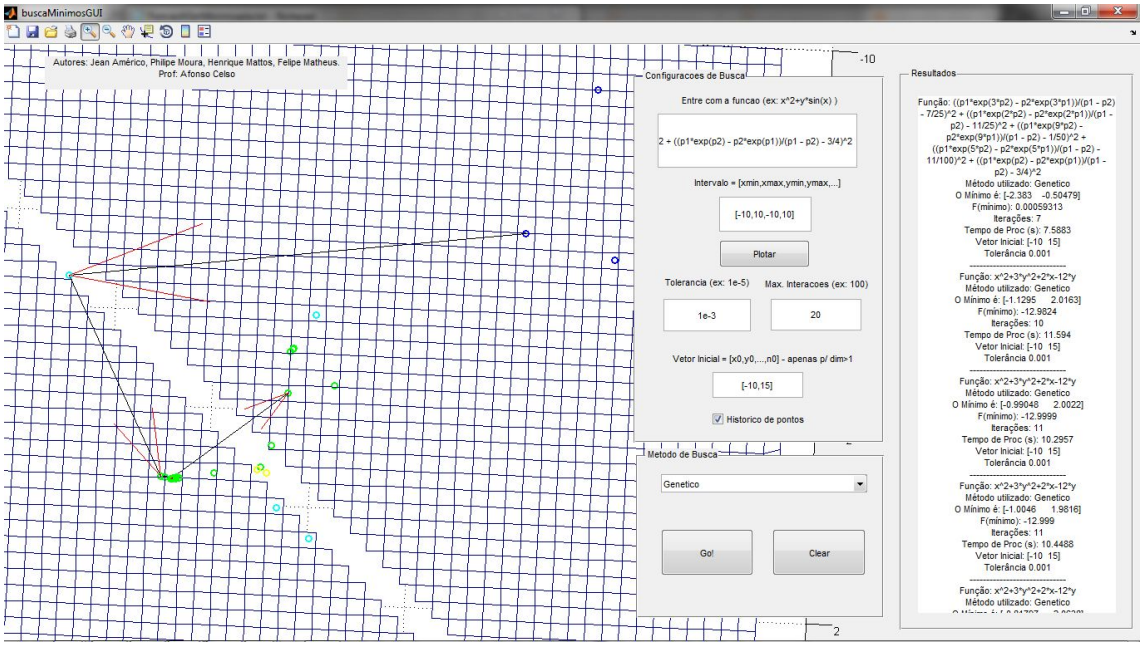

Figura 21: Interface do programa após a otimização achando $p_2 = -2.383$ e $p_1 = -0.50479$

6.3 Para 10 amostras

Pegando como pontos de amostragem todos os valores, obtemos a equação no formato:

```
/ p1 exp(3 p2) - p2 exp(3 p1)    7 \2   / p1 exp(6 p2) - p2 exp(6 p1)    2 \2   / p1 exp(2 p2) - p2 exp(2 p1)   11 \
\2
| ---------------------------- - -- |  + | ---------------------------- - -- |  + | ---------------------------- - -- |
\           p1 - p2             25 /    \           p1 - p2             25 /    \           p1 - p2             25 /

    / p1 exp(9 p2) - p2 exp(9 p1)    1 \2   / p1 exp(7 p2) - p2 exp(7 p1)    3  \2
  + | ---------------------------- - -- |  + | ---------------------------- - --- |
    \           p1 - p2             50 /    \           p1 - p2            100 /

    / p1 exp(8 p2) - p2 exp(8 p1)    3  \2   / p1 exp(5 p2) - p2 exp(5 p1)    11 \2
  + | ---------------------------- - --- |  + | ---------------------------- - --- |
    \           p1 - p2            100 /    \           p1 - p2             100 /

    / p1 exp(10 p2) - p2 exp(10 p1)    1  \2   / p1 exp(4 p2) - p2 exp(4 p1)    19 \2
  + | ----------------------------- - --- |  + | ---------------------------- - --- |
    \            p1 - p2              100 /    \           p1 - p2             100 /

    / p1 exp(p2) - p2 exp(p1)   3 \2
  + | ---------------------- - - |
    \         p1 - p2           4 /
```

Figura 22: Função representada no modo *pretty* do MATLAB

O resultado do programa em sua interface desenvolvida foi:

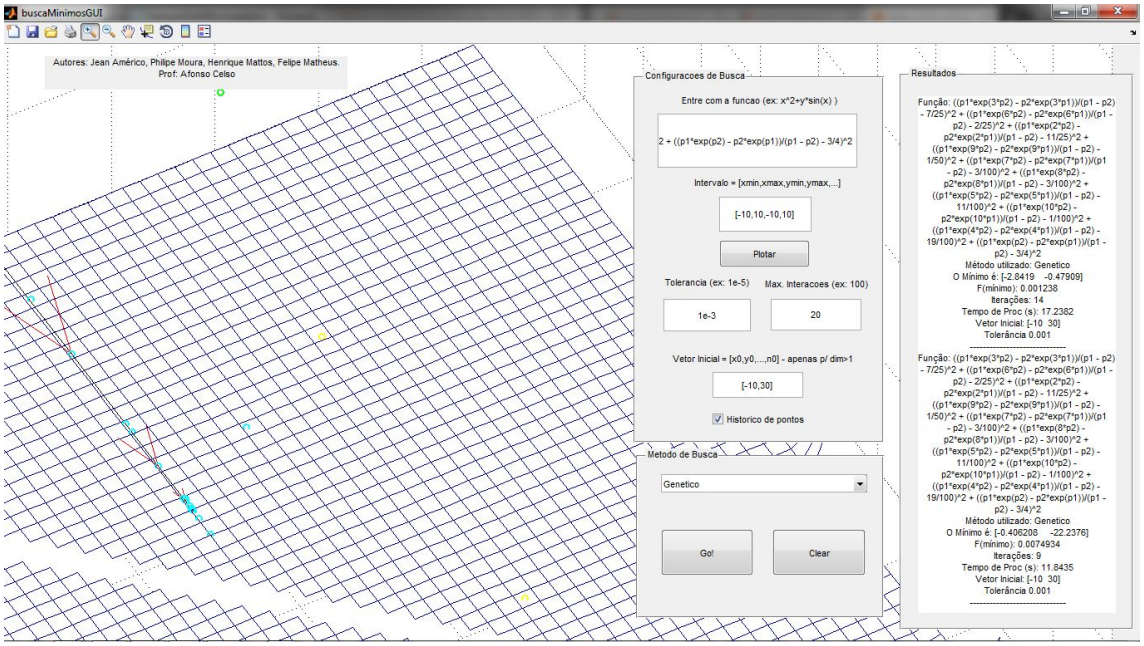

Figura 23: Interface do programa após a otimização achando $p_2 = -2.8419$ e $p_1 = -0.47909$

6.4 Resultado final

Por fim, tendo em vista que obtemos os valores de p_1 e p_2 para cada um dos 3 casos, o que nos fez chegar a 3 diferentes equações (pela substituição na equação 2), plotaremos cada equação substituindo t pelos valores de 1 até 10.

Assim, através da análise abaixo, percebemos que quanto maior o número de pontos de amostragem utilizados no método de mínimos quadrados, o gráfico final se enquadrará melhor aos pontos.

Para concluir, utilizamos os valores de p_1 e p_2 para o caso de 10 amostras nas equações (6) e (7), obtendo os valores de $\zeta = 1.42306$ e $\omega_n = 1.16684$.

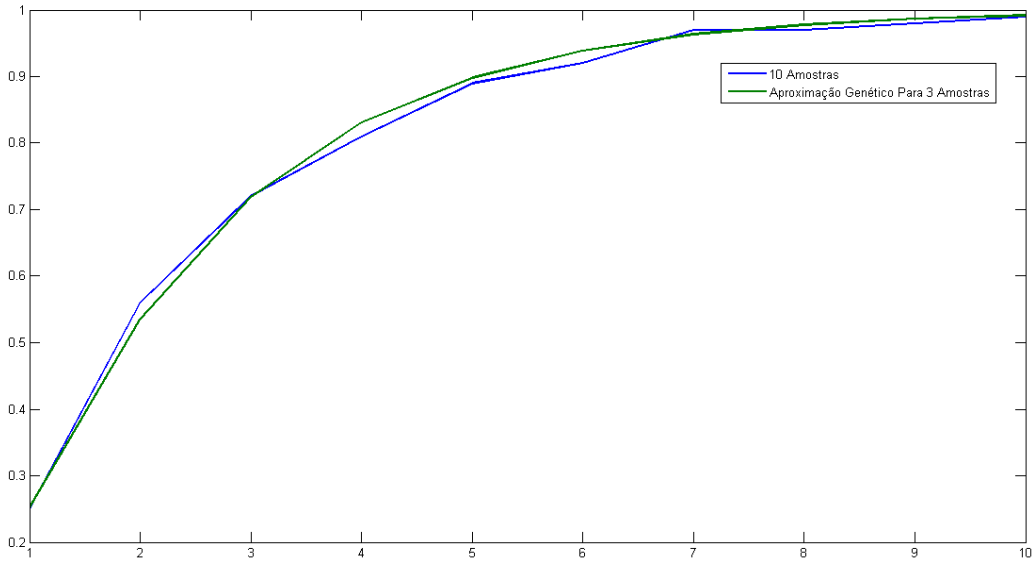

Figura 24: Função com os valores de p_1 e p_2 obtidos no 6.1

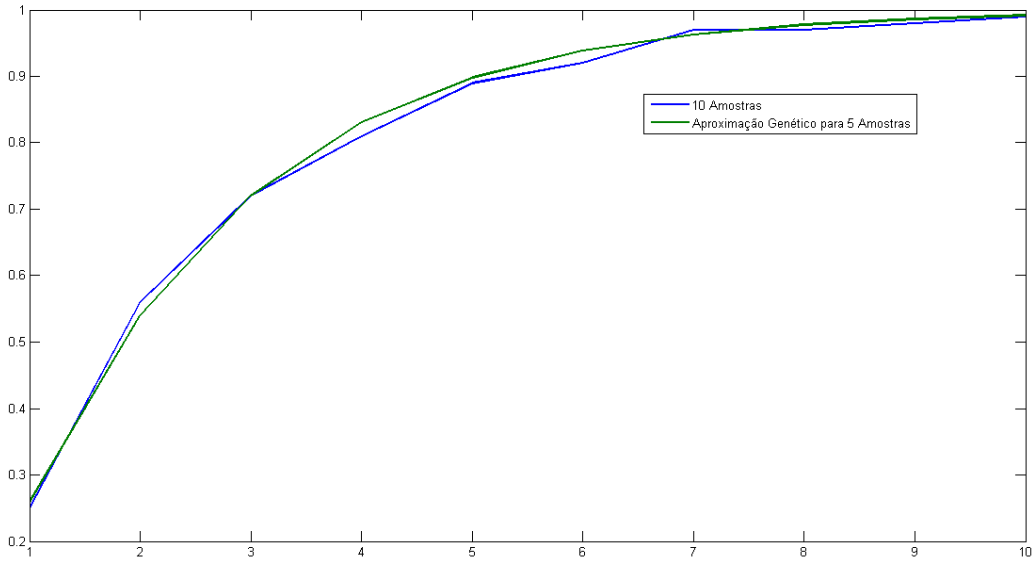

Figura 25: Função com os valores de p_1 e p_2 obtidos no 6.2

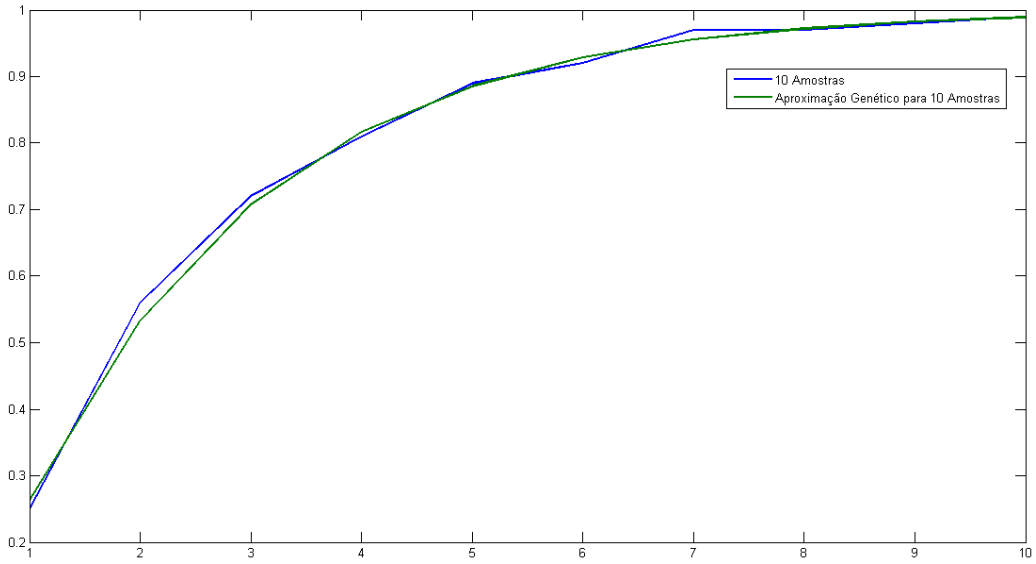

Figura 26: Função com os valores de p_1 e p_2 obtidos no 6.3

7 Interface do programa

Para a integração entre os métodos e a interação com os usuários, foi desenvolvida um interface através do *toolboox* GUIDE do MATLAB.

Nela, podemos, facilmente, selecionar o método de resolução desejada, a função, os parâmetros de busca e acompanhar o andamento do método através de vetores, tal como demonstra a figura 27.

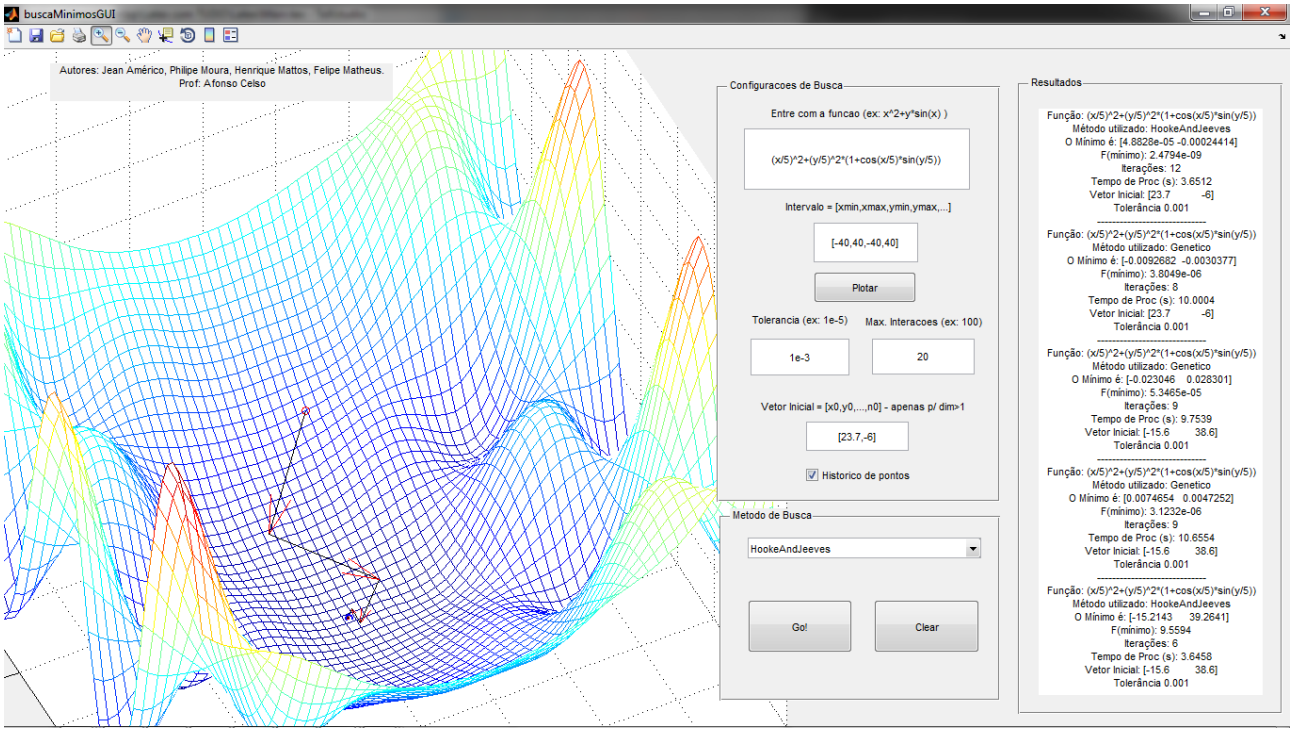

Figura 27: Interface gráfica desenvolvida através do GUIDE/MATLAB.

8 Conclusão

Após a análise das soluções alcançadas por ambos os métodos numéricos, podemos substituir na equação(1) e equação(2) os valores de $\zeta = 1.34942$, $\omega_n = 1.11231$, $p_2 = -2.5088$ e $p_1 = -0.49316$ pelo método de Hooke & Jeeves, obtendo as equações (9) e (10); e $\zeta = 1.42306$, $\omega_n = 1.16684$, $p_2 = -2.8419$ e $p_1 = -0.47909$ pelo método do algoritmo genético, obtendo as equações (11) e (12). Concluindo que as aproximações finais foram muito boas e conseguiram reduzir o erro encaixando se muito bem as amostras dos pontos, atendendo a expectativa da modelagem numérica da resposta ao degrau unitário do SLIT. Podemos conferir isso tanto pelas equações a seguir quanto pelos gráficos que comparam as amostras com a equação(2).

$$H(s) = \frac{1.2329}{s^2 + 2.99s + 1.2329} \tag{9}$$

$$h(t) = 1 + \frac{-2.5088e^{-0.49316t} + 0.49316e^{-2.5088t}}{2.01564} \tag{10}$$

$$H(s) = \frac{1.3615}{s^2 + 3.32s + 1.3615} \tag{11}$$

$$h(t) = 1 + \frac{-2.8419e^{-0.47909t} + 0.47909e^{-2.8419t}}{2.36281} \tag{12}$$

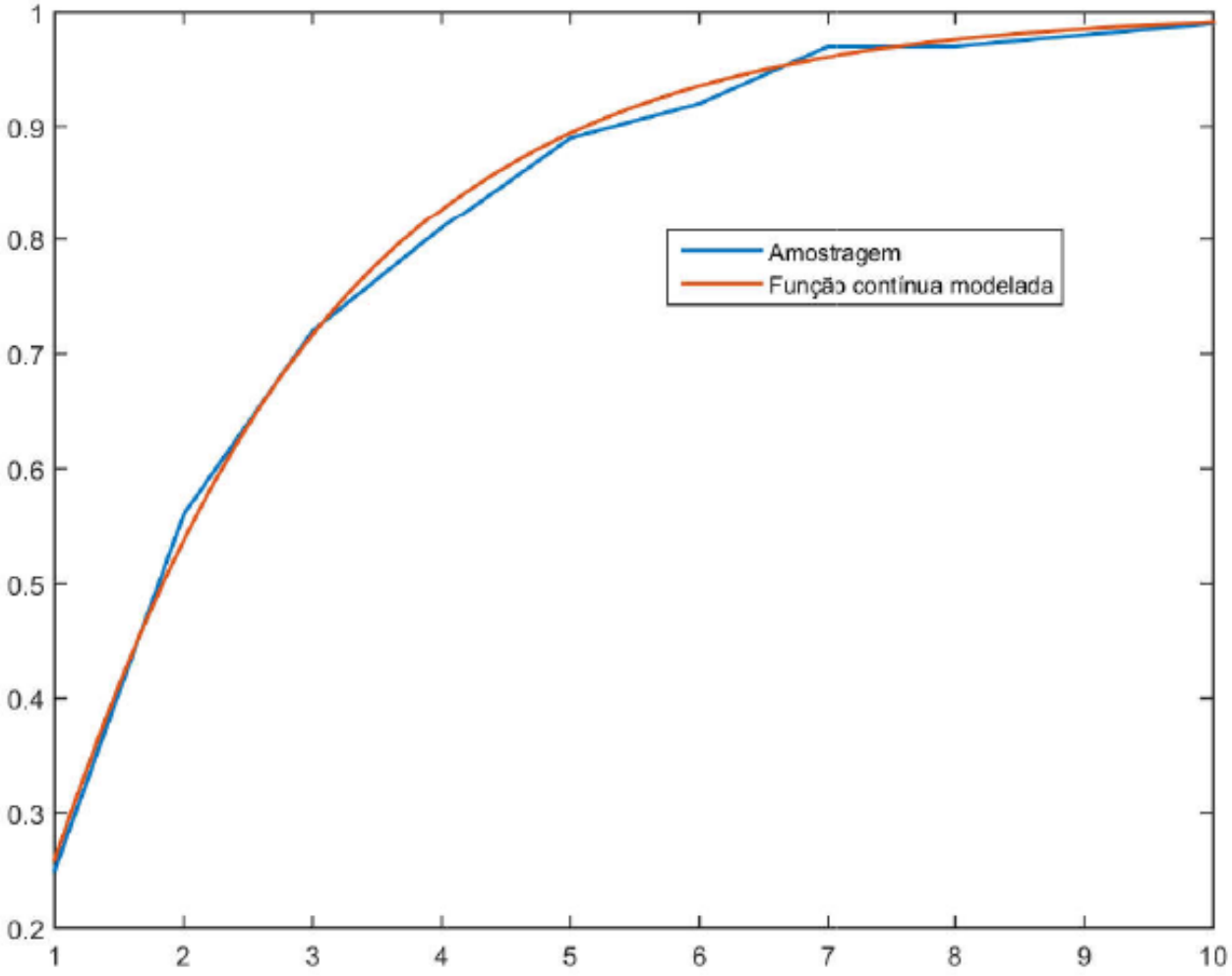

Figura 28: Comparação final pelo método de Hooke & Jeeves

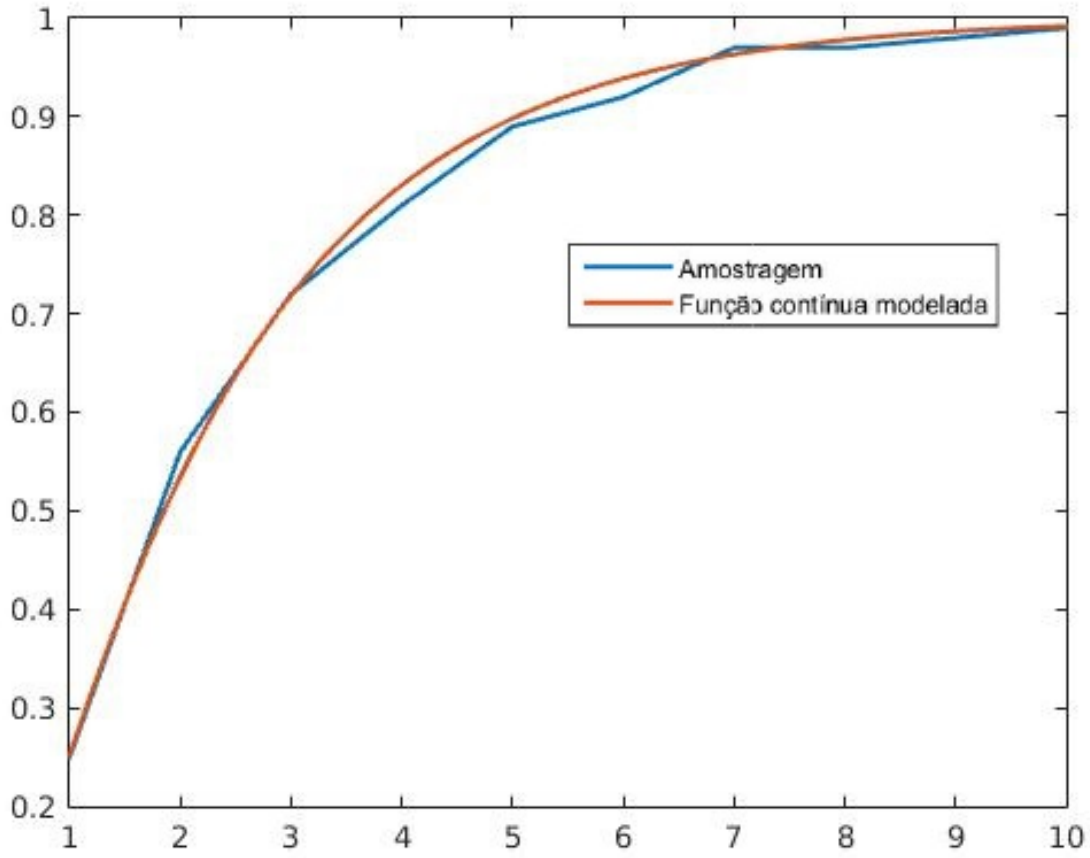

Figura 29: Comparação final pelo método do algoritmo genético

Referências

[1] J. H. Vuolo, *Fundamentos da teoria de erros.* Edgar Blucher, 1996.

[2] C. A. Perlingeiro, *Engenharia de Processos.* Blucher, 1976.

[3] A. Popov, *Genetic Algorithms for optimization*, 2005.

A Apêndice

A.1 Exploração

```
function [xMinimo,fMinimo, sucess] = HookeExploracao2(expr, base, incr)

    funcao = sym(expr); % funcao ja simbolica
    variaveis = symvar(funcao); % vetor com as variaveis ja simbolicas da funcao
    n1 = size(variaveis);
    n = n1(2); %quantidade de variaveis na funcao
%--------------------------------------------------------------------------
%   Exploracao
%--------------------------------------------------------------------------
    x = base;

    for i=1:n
        %------------------------------------ verifica para x(i)+incr
         novox = x;
         novox(i) = novox(i)+incr;
         if double(subs(funcao,variaveis,novox)) < double(subs(funcao,variaveis,x))
             x = novox;
             continue %se e menor de um lado, nao verifica do outro. Premissa: unimodalidade
         end
        %------------------------------------ verifica para x(i)-incr
         novox = x;
         novox(i) = novox(i)-incr;
         if double(subs(funcao,variaveis,novox))<double(subs(funcao,variaveis,x))
             x = novox;
         end
    end
    display({'incremento',num2str(incr)})

   if min(x==base)
       sucess = 0;
   else
       sucess =1;
   end

   xMinimo = x;
   fMinimo = double(subs(funcao,variaveis,x));
   %display({'Sucess',num2str(xMinimo~=base)})
```

A.2 Progressão

```
function [ ponto_min, valor_min, numI, historico] = Hooke(expr, x0, incr, tol, lim_it )
%--------------------------------------------------------------------------
%   Setup Inicial
%--------------------------------------------------------------------------
%incr = incremento
%tol = tolerancia, em forma de vetor linha -> cada coluna uma variavel
x = x0; % ponto inicial
numI = 0;
funcao = sym(expr); % funcao ja simbolica
variaveis = symvar(funcao); % vetor com as variaveis ja simbolicas da funcao
n1 = size(variaveis);
n = n1(2); %quantidade de variaveis na funcao

%--------------------------------------------------------------------------
%   Progressao
%--------------------------------------------------------------------------
%historico = horzcat(x0,double(subs(funcao,variaveis,x0)));
historico = [];
while max(incr)>tol && numI<lim_it
```

```
    disp({'x','f(x)',num2str(x),double(subs(funcao,variaveis,x))})

    [x1,fMin,sucess] = HookeExploracao2(expr,x,incr);  %procura o minimo ao redor da base
    historico = horzcat(historico,x,double(subs(funcao,variaveis,x)));
    %historico = horzcat(historico,x1,fMin);


    disp({'x1','f(x1)',num2str(x1),num2str(fMin)})
    sucess
    if sucess ==1
        x2 = 2*x1-x; %tentativa de progressao
        %disp({'tentativa de progressao', num2str(x2),double(subs(funcao,variaveis,x2))})
        %disp({'ponto atual', num2str(x1),fMin})

        numI = numI+1;
        while double(subs(funcao,variaveis,x2))<fMin
            historico = horzcat(historico,x2,double(subs(funcao,variaveis,x2)));
            disp({'progressao OK', num2str(x2),double(subs(funcao,variaveis,x2))})

            fMin = double(subs(funcao,variaveis,x2)); %f(x2) antigo. Para comparar com progressoes suscessi

            x1 = x2; %progressao antiga vira a a nova direcao de minimo
            x = x1; %a base agora e a direcao de minimo antiga
            x2 = 2*x1-x;   %fazendo mais uma progressao


        end
        if double(subs(funcao,variaveis,x2))>fMin
            disp({'progressao falhou',num2str(x2),double(subs(funcao,variaveis,x2))});
            x = x1; %caso contrario, continua em x1 p exploracao
        end

    else

        display('a base foi reduzida')


        incr = incr/2;
        x = x1;
    end


end

ponto_min = x1;
valor_min = fMin;
```

A.3 Geração de pontos

```
function [pop] = geraPontos(pontoInicial, tamanho)
%ponto inicial = [x,y]
dimensao = size(pontoInicial);
pop = [];
for k=1:dimensao(2)

    x1 = random('Normal',pontoInicial(k),5,tamanho,1); %para a determinada dimensao, gera "tamanho" pontos
%5.

    pop = cat(2,pop,x1);
end
```

A.4 Ordenamento da população

```
function [pop]=sortPop(pop)
tamanho=size(pop);
tamanho=tamanho(2);
pop=sortrows(pop,tamanho);
```

A.5 Seleção Natural

```
function [ output_args ] = selecao(matrixPontos,funcao, x0, n)
%A funcao retorna os n melhores pares de pontos (x,y) com os menores f(x,y)
%da matrix de pontos.
%   Detailed explanation goes here
%



end
```

A.6 Recombinação (crossover)

```
function [pontosRecombinados] = crossover(melhoresPontos,alpha)
%alpha = coeficiente de exploracao
%   O objetivo desas funcao e gerar todas combinacoes dois a dois de cada
%   vetor linha da dimensao K da matriz de melhoresPontos
dimensao = size(melhoresPontos);
pontosRecombinados = [];
colunaDeNovosPontos = [];
%------------- parametros aleatorios
r = random('Uniform',0,1);
gama = (1+2*alpha)*r-alpha;

%------------------ gera combinacao dois a dois de cada dimensao da matriz
for k=1:dimensao(2)
    disp('to aqui');
    combinacao = combntns(melhoresPontos(:,k),2); %combina o vetor linha da dimensao k
    %pontosRecombinatos = cat(pontosRecombinados,combinacao)

    %----------------------------

        sizeComb = size(combinacao);
        for kk=1:sizeComb(1)
            C1 = gama*combinacao(kk,1)+(1-gama)*combinacao(kk,2); %filho 1
            C2 = (1-gama)*combinacao(kk,1)+(1-gama)*combinacao(kk,2); %filho 2
            colunaDeNovosPontos = cat(1,colunaDeNovosPontos,[C1;C2]); %adiciona o vetor coluna [c1;c2] a co
        end



        pontosRecombinados = cat(2,pontosRecombinados,colunaDeNovosPontos); %adiciona o vetor coluna calcul
        colunaDeNovosPontos = [];
end

pontosRecombinados = cat(1,pontosRecombinados,melhoresPontos);

size(pontosRecombinados)
```

A.7 Mutação

```
function [popMutada] = mutacao(pop,beta)
%modulo de beta.
dimensao = size(pop);
i=1;
dimPop=size(pop);
dimPop=dimPop(2);
delta=[];
tamanhoPop=size(pop);
tamanhoPop=tamanhoPop(1);

vetorMutada = [];
popMutada = [];
%o while abaixo determina a diferenca entre o maximo e o minimo de cada
%dimensao (Xmax-Xmin, Ymax-Ymin...)

while i<=dimPop
    pop=sortrows(pop,i);
    deltaxi=(pop(tamanhoPop,i)-pop(1,i));  %delta da i-esima dimensao
    delta=[delta deltaxi];
    i=i+1;
end


for kk=1:dimensao(2)   %de kk=1 ate a quantidade de dimensoes
    for k=1:dimensao(1)  %de k=1 ate a quantidade de linha
        for y=1:2 %10 vezes para cada x
            alpha = random('Uniform',-beta,beta); %fator multiplicativo entre 0 e 0.5
            vetorMutada = cat(1,vetorMutada,pop(k,kk)+alpha*delta(1,kk));
         end

    end
    popMutada = cat(2,popMutada,vetorMutada);
    vetorMutada = [];
end
popMutada = cat(1,popMutada,pop);
size(popMutada)
```