Método do Formigueiro para Encontrar os Zeros de Funções Reais

Thiago da Silva Teixeira[1], **Iara da Cunha R. da Silva**[2]

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa, PR

[2]Departamento Acadêmico de Matemática
Universidade Tecnológica Federal do Paraná (UTFPR) – Ponta Grossa, PR

tteixeira@alunos.utfpr.edu.br, iarasilva@utfpr.edu.br

Abstract. *This article proposes a method that combines deterministic algorithms and a heuristic for find zeros of real functions, that is, find $x \in \mathbb{R}$ such that $f(x) = 0$. The methods converge to a single zero of the function f if hypotheses under it are satisfied, even if the function has more solutions. The heuristic method used in this work is the ant colony optimization that does not take into account the assumptions of the function f but analyzes it to determine which deterministic method to use. The results show the against of deterministic algorithms, the ant colony optimization return more than one solution, case the function has more real roots, and although it needs many iterations, the proposed strategy is capable of to find roots of discontinuous functions.*

Resumo. *Este artigo propõem um método que mescla algoritmos determinísticos e uma heurística para encontrar zeros de funções reais, ou seja, encontrar $x \in \mathbb{R}$ tal que $f(x) = 0$. Os métodos determinísticos precisam de hipóteses em relação a função f para convergirem e retornam apenas uma única solução real, mesmo que a função tenha mais soluções. A heurística utilizada neste trabalho é o método do formigueiro, que não leva em consideração as hipóteses da função f, mas faz uma análise nela para saber qual método determinístico que será utilizado. Os resultados deste artigo mostraram que diferentemente dos algoritmos determinísticos, o método do formigueiro retorna mais de uma solução real de f, caso a função tenha mais de um zero real, e apesar de necessitar de mais iterações, o método proposto é capaz de encontrar raízes de funções descontínuas.*

1. Introdução

Muitos problemas físicos podem ser modelados por uma equação de uma variável,

$$f(x) = 0 \tag{1}$$

onde $f : \mathbb{R} \longrightarrow \mathbb{R}$.

Um exemplo real deste problema é a função que descreve a velocidade de um paraquedista, deduzida a partir da segunda lei de Newton [Chapra and Canale 2009],

$$v(t) = \frac{gm}{c}(1 - e^{-(c/m)t}) \tag{2}$$

onde v é a variável dependente, o tempo t é a variável independente, a constante gravitacional g é o termo forçante e o coeficiente de arrasto c e a massa m são os parâmetros. Essa equação pode ser utilizada para determinar o coeficiente de arrasto para que um paraquedista de uma dada massa atinja uma certa velocidade em um determinado intervalo de tempo. A equação poderá ser reformulada

$$f(c) = \frac{gm}{c}(1 - e^{-(c/m)t}) - v. \tag{3}$$

O valor c tal que $f(c) = 0$ é chamado de raiz da função (3) e representa também o coeficiente de arrasto que soluciona o problema físico.

Existem vários métodos para encontrar raízes de funções, esses métodos podem ser divididos em determinísticos e não determinísticos. Os determinísticos precisam de hipóteses para que os métodos convirjam, entre as hipóteses, um bom comportamento da função f é necessário [Burden and Faires 2011, Franco 2006, Ruggiero and da Rocha Lopes 1996]. Já os métodos heurísticos são construídos sem uma fundamentação teórica matemática e podem ser aplicados em função que não precisam satisfazer hipóteses como ser contínua e suave.

Desse modo, foi proposto um método que mescla métodos determinísticos e uma heurística, o método do formigueiro, que não leva em consideração o comportamento da função que se pretende encontrar a raiz, tratando-a como se fosse uma caixa preta. O funcionamento do método é análogo a uma colônia de formigas [Glover 2003] que irá gerenciar a aplicações dos métodos determinísticos, onde todas as configurações e parâmetros para convergir, caso necessário, passem para o formigueiro.

Este artigo está estruturado da seguinte forma. Na seção 2, descreveremos o método do formigueiro implementada neste trabalho. Na seção 3, apresentaremos os métodos determinísticos utilizados conjuntamente com o método do formigueiro. Na seção 5 temos a análise dos resultados dos testes realizados.

2. O Método do Formigueiro

O método do formigueiro tem como objetivo imitar alguns padrões da natureza relacionados a uma colônia de formigas. O método do formigueiro surgiu pela necessidade de um sistema lidar com números pseudo-aleatórios de forma controlada e sistemática, sendo capaz de manejar a aleatoriedade e todos seus benefícios sem um custo computacional que tenda ao infinito, possuindo parâmetros para reforçar ações e os critérios de parada [Glover 2003].

Quando o formigueiro é inicializado, este possui uma localização x_i, a população inicial pop_i e a área explorada A_i. A área explorada é dividida em intervalos chamados de setores s_k, de acordo com o número de formigas, cada formiga f_i é associada a um setor onde será colocada aleatoriamente no intervalo do setor associado, assim sucessivamente até que toda área explorada seja populada. A formiga interage com o ambiente através de p passos que tem a função de convergir a um objetivo determinado na aplicação desta estratégia. Quando todas as formigas realizam o máximo de passos p_{max} temos uma iteração do formigueiro chamada de geração g.

Se ao menos uma formiga f_i conseguir chegar em seu objetivo, então o formigueiro irá se alimentar expandindo para a próxima geração g_{i+1}, a pop_i será multipli-

cada por uma taxa de expansão t_{ex}, caso contrário o formigueiro irá contrair-se com a multiplicação de pop_i por uma taxa de contração t_{con}. Esta relação caracteriza o sucesso ou o fracasso de uma geração, a área explorada cresce com taxa t_{con} a cada geração, o algoritmo termina quando $pop <= 0$.

O algoritmo do método do formigueiro pode ser visto na seção 4.

3. Métodos Numéricos

Os métodos apresentados a seguir são a base para o comportamento das formigas, são usados para dar o passo das formigas em direção a raiz, cada método possui seus prós e contras, as formigas analisam a situação e escolhem qual método usar em busca de uma melhor convergência.

3.1. Método da Bisseção

O método da bisseção usa o Teorema de Bolzano [Lima 1999] que diz que se uma função real f definida em um intervalo contínuo $[a, b]$ e $f(a)f(b) < 0$, então existe pelo menos um $c \in]a, b[$, tal que $f(c) = 0$, essas hipóteses garantem a convergência do método. Obtendo os pontos a e b que satisfazem o Teorema de Bolzano é calculado um valor médio $m_n = \frac{a+b}{2}$ da n-ésima iteração, m_n deve substituir um dos valores a ou b de forma a manter a raiz no intervalo. Repete-se o processo até convergir, ou seja, até que o erro absoluto $e_n = |b-a|$ seja menor que a tolerância tol. O método possui convergência linear o que o torna muito lento, porém o fato de garantir a convergência realizando poucos cálculos por iterações o torna conveniente dependendo do problema.

3.2. Interpolação Quadrática Inversa

A interpolação quadrática [Epperson 2007] é usada para a aproximação de raízes não lineares onde sua definição parte da relação de recorrência. Para obter o polinômio de segunda ordem utiliza-se três pontos x_{n-2}, x_{n-1} e x_n. A interpolação quadrática é definida como:

$$f^{-1}(y) = \frac{(y-f_{n-1})(y-f_n)}{(f_{n-2}-f_{n-1})(f_{n-2}-f_n)}x_{n-2} + \frac{(y-f_{n-2})(y-f_n)}{(f_{n-1}-f_{n-2})(f_{n-1}-f_n)}x_{n-1} + \dots + \frac{(y-f_{n-2})(y-f_{n-1})}{(f_n-f_{n-2})(f_n-f_{n-1})}x_n \quad (4)$$

onde $f_i = f(x_i)$.

O método consegue convergir para a raiz de forma muito rápida, porém necessita de três pontos para iniciar o algoritmo e se os mesmos estiverem longe da raiz o desempenho cairá drasticamente. Por esta razão a interpolação quadrática inversa é combinada com outros métodos como o método de Brent (que pode ser visto em 3.4), ou usada em casos específicos, e sua ordem de convergência é aproximadamente 1.8 [Chapra and Canale 1973].

3.3. Método da Secante

O método da secante consiste em um variante do método de Newton com uma grande vantagem de não precisar do cálculo da derivada em sua iteração, sendo substituída por diferenças finitas.

Caso o método da secante convirja, sua ordem de convergência não é linear ou quadrática como no método de Newton, mas sim superlinear ≈ 1.618 [Burden and Faires 2011, Franco 2006].

Usa-se o método de Newton para a obtenção do método da secante:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{5}$$

Ao realizar a substituição da derivada por uma diferenças finitas obtemos:

$$f'(x_k) \cong \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}, \tag{6}$$

realizando as substituições necessárias chega-se na versão interativa:

$$x_{k+1} = \frac{x_{k-1}f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})}. \tag{7}$$

Nota-se que não há mais o cálculo da derivada e a obtenção do novo x_{k+1} necessita da avaliação da função por meio de dois pontos, x_k e x_{k-1}, sendo necessário dois pontos iniciais para o método da secante, ao contrário do método de Newton que usa somente um ponto, x_k.

3.4. Método de Brent

O método de Brent, também conhecido como método Dekker-Brent [Chapra and Canale 2009] trata-se de um algoritmo que usa a interpolação quadrática inversa e o método da secante para convergir mais rápido para raiz, e caso encontre algum problema, ele volta para o método da bisseção garantindo a convergência. A ideia de combinar o método da secante e o método da bisseção veio de Dekker. O método da bisseção tem a convergência garantida se dados dois pontos x_1 e x_2 tem-se $f(x_1)f(x_2) < 0$ e f é contínua em $[x_1, x_2]$, por outro lado o método da secante apresenta uma melhor convergência que o método da bisseção, devido a este resultado: seja x_k o chute inicial para raiz e a_k um outro ponto diferente de x_k, se $|f(x_k)| \leq |f(a_k)|$, então x_k é um melhor chute do que a_k.

O método de Dekker calcula duas aproximações s através do método da secante: se $f(x_k) \neq f(x_{k-1})$ então

$$s = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \tag{8}$$

senão

$$s = m \tag{9}$$

e m atráves do método bisseção:

$$m = \frac{a_k + x_k}{2}. \tag{10}$$

Se o resultado da secante, s, estiver entre x_k e m então $x_{k+1} = s$, senão atribui-se o valor obtido pelo método da bisseção $x_{k+1} = m$. Se $f(a_k)f(x_{k+1}) < 0$ então o

valor de a_k se mantém constante $a_{k+1} = a_k$, e se $f(x_{k+1})f(x_k) < 0$ então $a_{k+1} = x_k$. Consequentemente, se $|f(a_{k+1}| < |f(x_{k+1})|$, então a_{k+1} é um melhor chute que x_{k+1}.

O método de Dekker converge de forma eficiente, porém em certos casos, realiza mais iterações do que método da bisseção sozinho. Brent trouxe uma inovação para o método realizando um teste de quando o método da secante será aceito para a próxima iteração, duas comparações devem ser satisfeitas de acordo com uma tolerância δ. Se a iteração anterior usou a bisseção e $|\delta| < |x_k - x_{k-1}|$ então utiliza-se a interpolação, caso contrário, o método da bisseção é aplicada e seu resultado é usado na próxima iteração. Se a iteração anterior usou a interpolação e $|\delta| < |x_{k-1} - x_{k-2}|$, então aplica-se a interpolação novamente, caso contrário, é aplicado o método da bisseção. Se na iteração anterior foi utilizada o método da bisseção, então a condição $|s - x_k| < \frac{1}{2}|x_k - x_{k-1}|$ é satisfeita, senão utiliza-se o método da bisseção para a próxima iteração. Se a iteração anterior utilizou-se a interpolação então $|s - x_k| < \frac{1}{2}|x_{k-1}x_{k-2}|$ deve ser satisfeita.

Esta modificação garante que a iteração k, o método da bisseção será realizada em, no máximo $2\log_2(|x_{k-1} - x_{k-2}|/\delta)$ iterações adicionais, a condição citada força o uso da interpolação reduzindo pela metade a cada duas iterações até que o tamanho do passo seja menor que δ. Brent provou que o seu método requer no máximo N^2 iterações, em que N indica o número de iterações para o método da bisseção. Se a função f for contínua, então o método de Brent utilizará a interpolação quadrática ou linear inversa, nesse caso a convergência é superlinear [Chapra and Canale 1973].

4. Implementação do Método

O algoritmo foi implementado na linguagem python por ser uma linguagem de alto nível focada em simplicidade na sintaxe do código. Foram utilizados dois pacotes de programação científica: o Numpy para otimizar o desempenho do código e o PyROOT um framework que possui uma implementação robusta do método de Brent.

Os resultados dos testes apresentados na seção 5 apresentam dois usos possíveis para o formigueiro, um encerrando o algoritmo após a obtenção de uma raiz e outro que busca ao menos uma raiz e se alimenta através das raízes entrando em expansão ou declínio de acordo com o sucesso obtido pela geração definido na tabela 1.

O formigueiro foi configurado da seguinte forma:

```
max_int = 50
inicio = 0
tamanho = 100
max_tamanho = 1000
populacao = 2
max_populacao = 1000
min_raizes = 1
max_iteracao = 50000
```

onde

max_int: número máximo de iteração de cada formiga por geração;
repelente: índice de arredondamento re, quando um passo de valor obtido x, e o formigueiro possuir um ou mais raízes r se $x - 10^{-re} < r < x + 10^{-re}$ obter um novo

```
def ant(f, x1, x2, erro = 0.001):
        f1 = f(x1)
        f2 = f(x2)

        iteracao = 0

        if f1*f2 < 0:
                saida = metodo_brent(f,x1,x2,erro)
                if saida[0]:
                        return saida[1], saida[2]
                else:
                        iteracao += saida[1]

        try:
                novo = ( x1 * f2 - x2 * f1 ) / ( f2 - f1 )
        except ZeroDivisionError:
                return x2, iteracao+1

        return novo, iteracao+1
```

Tabela 1. Definição de geração de formigas.

valor para a formiga expulsando-a para longe da raiz evitando aproximações já calculadas;

inicio: ponto de localização do formigueiro x_i;

tamanho: tamanho tam para determinar a área explorada onde $A_i = [x_i - tam, x_i + tam]$;

max_tamanho: máximo valor atribuído à variável tamanho;

populacao: quantidade de formigas pop_i;

max_populacao: máximo número de formigas;

min_raizes: número mínimo de raízes encontradas pelo formigueiro antes de começar seu declínio;

max_iteracao: número máximo de iterações de todas as gerações.

5. Resultados Computacionais e Análise

O algoritmo do método do formigueiro implementado neste trabalho foi aplicado em quatro funções de uma variável e seus resultados comparados com os dos métodos determinísticos: bisseção, Newton e secante.

Para inicializar os testes, foi utilizado uma função polinomial de quarta ordem, $f_1(x) = x^4 - 16$, com raízes em -2 e 2, cada uma com multiplicidade 2. Por ser uma função contínua e suave todos os métodos convergiram. Os métodos determinísticos, retornaram apenas uma solução, já a heurística do método do formigueiro retornou as duas raízes desta função. Na tabela 2, tem-se o desempenho e as soluções alcançadas de cada método.

A segunda função utilizada nos testes, também é contínua, com infinitos zeros reais, como pode ser visto na figura 1. Os três métodos determinísticos convergiram para

Tabela 2. $f_1(x) = x^4 - 16$

	Solução ($\bar{x}$)	$f(\bar{x})$	Nº de iterações
Formigueiro	-2	0,00000	31
	-2	0,00000	34
Bisseção	-1,99999	-0,00024	16
Newton-Raphson	2	0,00000	6
Secante	2	0,00000	26

soluções diferentes, sendo o método de Newton com o melhor desempenho para esse caso. O método do formigueiro retornou 2466 possíveis soluções, sendo as melhores com erro de próximo de 0,02. Os resultados para cada método podem ser conferidos na tabela 3.

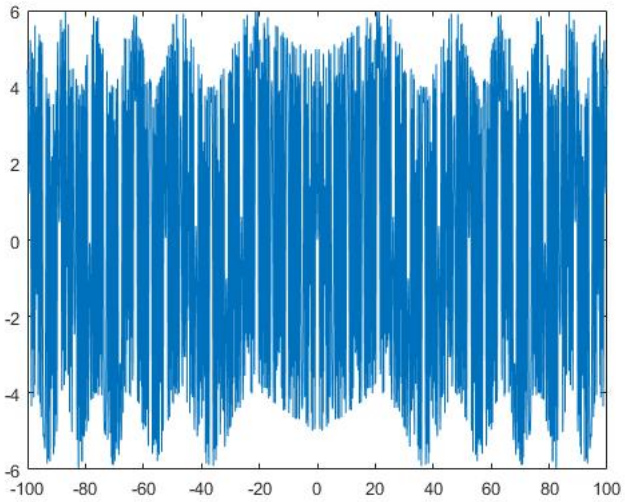

Figura 1. Gráfico da função $f(x) = \sin(0,0035x^2) + 5\sin(6x^2)$**.**

Tabela 3. $f(x) = \sin(0,0035x^2) + 5\sin(6x^2)$

	Solução ($\bar{x}$)	$f(\bar{x})$	Nº de iterações
Formigueiro	-23,96700	0,03419	300
	11,06800	-0,22716	331
	-63,26400	-0,63830	470
	⋮	⋮	⋮
Bisseção	-2,80261	0,00375	17
Newton-Raphson	3,15429	0,00006	2
Secante	2133,43	-0,00005	48

A terceira e quarta funções, sendo respectivamente, $f_3(x) = \dfrac{2x^2 + 3x - 9}{x^2 - 2x}$ e $f_4(x) = \tan\left(\dfrac{1}{x}\right)$, apresentam descontinuidades, a primeira em $x = 0$ e a segunda

função nos pontos $x = 0$ e $x = 2$. Os métodos determinísticos citados neste trabalho garantem convergência, caso hipóteses sejam satisfeitas [Burden and Faires 2011], dentre elas que a função seja contínua. Observe na tabela 4, que apesar das 2470 iterações, o método do formigueiro convergiu para a raiz da função $f_3(x)$, ao contrário dos métodos determinísticos, que tem como hipóteses para convergirem a continuidade da função real.

Tabela 4. $f(x) = \dfrac{2x^2 + 3x - 9}{x^2 - 2x}$

	Solução ($\bar{x}$)	$f(\bar{x})$	Nº de iterações
Formigueiro	-3	0,00000	2470
Bisseção	Não convergiu		
Newton-Raphson	Não convergiu		
Secante	Não convergiu		

A função $f(x) = \tan\left(\dfrac{1}{x}\right)$ possui infinitas raízes, $x^* = \frac{1}{\pi n}, n \in \mathbb{Z}^*$. Como pode ser observado na tabela 5, o método do formigueiro retornou 9 soluções, sendo duas delas (iterações 802 e 3638) com erro de magnitude próximo a 0,1. O único método determinístico que convergiu foi o de Newton-Raphson, pois foi utilizado um ponto inicial bem próximo de uma das raízes, qualquer outro ponto inicial longe das raízes, foi verificado empiricamente que o método de Newton-Raphson não convergiu.

Tabela 5. $f(x) = \tan\left(\dfrac{1}{x}\right)$

	Solução ($\bar{x}$)	$f(\bar{x})$	Nº de iterações
Formigueiro	0,15900	0,00612	40
	-0,31800	-0,00306	59
	0,31800	0,00306	63
	-0,05300	-0,01837	169
	0,10600	0,00918	598
	0,00800	-0.78206	802
	-0,15900	-0,00612	1619
	0,04500	0,23528	3638
	-0,10600	-0,00918	3877
Bisseção	Não convergiu		
Newton-Raphson	0,31831	0,00000	5
Secante	Não convergiu		

6. Conclusão

Diferentemente dos métodos determinísticos, o método do formigueiro não precisa de hipóteses e nenhuma outra informação a respeito da função que se queira encontrar as raízes. Os métodos determinísticos além das hipótese para convergirem, precisam de informações de intervalos que contenham pelo menos uma raiz, pontos inicias ou até mesmo informação da derivada, como é o caso do método de Newton-Raphson.

O método do formigueiro apesar de ser custosa traz uma nova abordagem para a resolução de equações $f(x) = 0$, podendo encontrar várias raízes e não necessitando

que o usuário faça algum tipo de gráfico ou análise da função para usar o método. Outra vantagem é caso se queira aprimorar o método de Brent, o algoritmo pode ser facilmente adaptado modificando a função ant(), mantendo a usabilidade do formigueiro.

Referências

Burden, R. L. and Faires, J. D. (2011). *Numerical Analysis*. Cengage Learning, 9 edition.

Chapra, S. and Canale, R. (1973). *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ: Prentice-Hall.

Chapra, S. and Canale, R. (2009). *Métodos Numéricos para Engenharia - 5.ed.:*. McGraw Hill Brasil.

Epperson, J. F. (2007). *An introduction to numerical methods and analysis*. Wiley-Interscience.

Franco, N. B. (2006). *Cálculo numérico*, volume 1. São Paulo: Pearson Prentice Hall, 1 edition.

Glover, Fred; Kochenberger, G. A. (2003). *[International Series in Operations Research & Management Science] Handbook of Metaheuristics Volume 57 —— The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances*, volume 10.1007/b101874.

Lima, E. L. (1999). *Numerical mathematics*, volume 1. IMPA.

Ruggiero, M. A. and da Rocha Lopes, V. L. (1996). *Cálculo numérico: aspectos teóricos e computacionais*. Makron Books do Brasil.