Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 2, N. 1, 2014.
Trabalho apresentado no CMAC-Sul, Curitiba-PR, 2014.

Aplicação do Sistema de Girard no Cálculo de Raízes de Polinômios

Oliver Kolossoski
Programa de Pós-Graduação em Matemática Aplicada,
PPGMA - Centro Politécnico, 81530-900 Curitiba, Pr, Brazil
email: OliverKolossoski@gmail.com

Luiz Carlos Matioli
Departamento de Matemática,
Av. Cel Francisco H dos Santos 81530-900 Curitiba, Pr, Brazil.
email: matioli@ufpr.br

Resumo: *Em nosso trabalho introduzimos uma metodologia que transforma um sistema de somas de produtos de potências em um sistema de somas de apenas potências. Uma vez que as relações de Girard estão relacionadas com os zeros de um polinômio e essas relações definem um sistema de Girard, que é um sistema de somas de produtos de potências, utilizamos nossa metodologia para transformar o sistema de Girard em um sistema equivalente, com apenas somas de potências. O sistema equivalente gerado é mais fácil de ser trabalhado, no sentido de procurar sua solução. Assim, aplicamos o método de Newton para resolver o sistema equivalente, a fim de tirar proveito de sua estrutura. Exemplificamos nossa metodologia calculando todas as raízes de um polinômio, desde que sejam distintas.*

1 Introdução

O problema de se calcular todas as raízes de um polinômio dado, da forma $p(x) = \sum_{j=0}^{n} a_j x^{n-j} = \prod_{j=1}^{n}(x - \alpha_j)$ com $a_0 = 1$, e onde α_j, $j = 1, \ldots n$ são as raízes de $p(x)$, é bastante conhecido, e existem diversos métodos numéricos desenvolvidos com tal finalidade. No entanto, dado um polinômio com n raízes distintas, são poucos os métodos que exibem uma sequência de n pontos que convergem para suas n raízes. Em geral, os métodos calculam

DOI: 10.5540/03.2014.002.01.0063

as raízes de duas em duas, efetuando o processo de deflação e aplicando novamente o algoritmo para um polinômio de grau $n-2$.

O objetivo deste trabalho é desenvolver um método que exibe tal sequência. O método é baseado no sistema de equações de Girard, o qual mostramos ser equivalente a um sistema de somas de potências. Mais precisamente, transformamos o seguinte sistema de Girard de n equações

$$\left\{\begin{array}{ccccccccccc} x_1 & + & x_2 & + & x_3 & \dots & + & x_n & = & -a_1 \\ & & x_1x_2 & + & x_1x_3 & \dots & + & x_{n-1}x_n & = & a_2 \\ & & & & \vdots & & & \vdots & \vdots & \vdots \\ & & & & & & & x_1x_2\dots x_n & = & (-1)^n a_n \end{array}\right. \tag{1}$$

em um sistema de equações equivalente, com apenas somas de potências, a saber:

$$\left\{\begin{array}{lcl} x_1 + x_2 + \cdots + x_n & = & c_1 \\ x_1^2 + x_2^2 + \cdots + x_n^2 & = & c_2 \\ \vdots & & \vdots \\ x_1^n + x_2^n + \cdots + x_n^n & = & c_n \end{array}\right. \tag{2}$$

2 O Método

Na primeira parte de nosso trabalho, procuramos obter uma expressão para os coeficientes c_i a serem encontrados no sistema (2) Após diversas manipulações algébricas, encontramos uma expressão recursiva para os coeficientes c_k no sistema (2). Tais relações estão dadas no teorema a seguir:

Teorema 1 *Dado o sistema de Girard de n equações (1), os coeficientes c_i do sistema (2) podem ser computados via*

Algoritmo 1 $c_1 = -a_1$
Para $k = 1, \dots, n$

$$c_k = -\sum_{j=1}^{k-1} a_j c_{k-j} - k a_k.$$

End

Como aplicação da fórmula dada no algoritmo 1, podemos derivar um método que encontra todas raízes de um dado polinômio de uma vez só, desde que sejam distintas.

A ideia é usar o método de Newton no sistema (2) para achar uma sequência de vetores $[x_1^k, x_2^k, \dots x_n^k]$ que converge para um vetor $[x_1, x_2, \dots x_n]$

DOI: 10.5540/03.2014.002.01.0063

com todas as n raízes. Uma vez encontrado o sistema equivalente, podemos aplicar o método para encontrar suas raízes: Efetuamos os testes usando o método de Newton puro, isto é, sem busca, para ilustrar a metodologia. No entanto, pode ser usada uma busca para melhoria de resultados ou um método quasi-Newton, para acelerar o processo. A iteração é dada por:

$$x^{(k+1)} = x^{(k)} - (F'(x^{(k)}))^{-1} F(x^{(k)})$$

onde $F(x)$ é dada por

$$F(x) = \begin{pmatrix} x_1 + x_2 + \cdots + x_n - c_1 \\ x_1^2 + x_2^2 + \cdots + x_n^2 - c_2 \\ \vdots \\ x_1^n + x_2^n + \cdots + x_n^n - c_n \end{pmatrix} \tag{3}$$

e $F'(x^{(k)})$ é a matriz jacobiana de $F(x)$ em $x^{(k)}$ dada por

$$F'(x) = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 2x_1 & 2x_2 & \dots & 2x_n \\ & \vdots & & \\ nx_1^{n-1} & nx_2^{n-1} & \dots & nx_n^{n-1} \end{bmatrix}, \tag{4}$$

o que acarreta numa matriz de Vandermonde de ordem n. Tem-se, assim, o algoritmo

ALGORITMO 2 *Dados $x^{(0)}$ valor inicial para todas as raízes de $p(x)$, $\epsilon > 0$ e $k = 0$.*
enquanto $||F(x^{(k)})|| \geq \epsilon$ *faça*
$F'(x^{(k)})d^{(k)} = -F(x^{(k)})$
$x^{(k+1)} = x^{(k)} + d^{(k)}$
$k = k + 1$
Fim

com $F(x^{(k)})$ dada por (3) e $F'(x^{(k)})$ por (4).

Pode-se resolver o sistema de Vandermonde em $O(n^2)$ (ver [3]), ou ainda, como a solução do sistema depende apenas de c_i, que por sua vez dependem apenas dos coeficientes do polinômio, pode-se computar a solução algébrica do sistema para os coeficientes a_i e incorporar a solução na iteração, reduzindo o custo para apenas $O(n)$ flops por iteração.

DOI: 10.5540/03.2014.002.01.0063

3 Experimentos Numéricos

Efetuamos experimentos aplicando nossa metodologia a diversos polinômios, alguns deles representados na Tabela 1. A precisão aqui utilizada foi 10^{-4}, e o critério de parada, a norma do vetor $F(x^{(k)})$, dado pela expressão (3). Os polinômios usados na Tabela 1 são:
$P_1(x) = x^3 - 6x^2 + 11x - 6$, com raízes $1, 2$, e 3.
$P_2(x) = x^4 - 80x^2 + 1024$, com raízes $4, -4, 8$ e -8.
$P_3(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$, com raízes $1, 2, 3$ e 4.
$P_4(x) = x^4 - 2x^3 - x^2 + 2x$, com raízes $0, -1, 1$ e 2.
$P_5(x) = x^4 - 2x^3 - 2x^2 + 8x - 8$, com raízes $1 + i, 1 - i, 2$ e -2.
$P_6(x) = x^4 - 0.18x^3 + 0.0119x^2 - 0.000342x + 0.0000036$, com raízes $0.03, 0.04, 0.05$ e 0.06.
$P_7(x) = x^4 - 10x^3 + 37x^2 - 60x + 36$, com raízes $2, 2, 3$ e 3.

$p(x)$	valor inicial	iter	vetor encontrado
P_1	$[4; 5; 6]$	15	$[3 + 10^{-7}; 1; 2 - 10^{-8}]$
	$[10^9; 10^6; 10^-4]$	52	$[3; 2 + 3 \times 10^{-8}; 1 - 3 \times 10^{-8}]$
	$[1.1; 2.2; 3.3]$	4	$[1; 2; 3]$
	$[1; 2; 3004]$	28	$[1; 3; 2]$
P_2	$[1; 2; 3; 5]$	620	$[4; 8; -8; -4]$
	$[1; 2; 3; 4]$	90	$[8; -8; 4; -4]$
	$[7.8; -8.2; -3.78; 4.15]$	3	$[8; -8; -4; 4]$
P_3	$[8; 5; 6; 7]$	249	$[2; 1 + 10^{-9}; 3; 4]$
	$[4; 5; 6; 7]$	1515	$[4; 1; 3; 2]$
	$[0; 5; 6; 9]$	46	$[2 - 8 \times 10^{-7}; 3 - 8 \times 10^{-7}; 4 - 4 \times 10^{-7}; 1 - 4 \times 10^{-7}]$
	$[1.1; 1.9; 2.85; 4.22]$	4	$[1; 2; 3; 4]$
	$[1.0001; 2.0002; 3.0003; 4.0004]$	1	$[1 + 10^{-9}; 2 + 10^{-8}; 2 - 2 \times 10^{-7}; 4 + 2 \times 10^{-7}]$
	$[6.; 9; 14; 2.621^{11}]$	2741	$[1 - 2 \times 10^{-7}; 3 + 2 \times 10^{-7}; 4 + 2 \times 10^{-7}; 2 - 2 \times 10^{-7}]$
	$[1 + i; 2 + i; 8 + i; 5 + i]$	9	$[2 - 3.469^{-10}i; 3 + 1.728^{-10}i; 4 - 6.621^{-12}i; 1 + 1.807^{-10}i]$
P_4	$[3; 4; 5; 6]$	648	$[1; 3.229^{-11}; -1; 2]$
	$[0; 4; 5; 6]$	32	$[2 + 1.6 \times 10^{-6}; 1 - 2 \times 10^{-6}; 2 \times 10^{-6}; -1 + 1.6 \times 10^{-6}]$
P_5	$[1; 3 + i; 4 + i; 5]$	12	$[1 - i; 2 - 10^{-8} - 4.477^{-8}i; -2 + 2.941^{-11}i; 1 + i]$
	$[i; 1 + i; 4; 2.23]$	9	$[1 - i; 1 + i; 2 - 2.759^{-10}i; -2 - 6.441^{-12}i]$
	$[-i; 1.1 + i; 2i; -3.]$	11	$[2 + 1.190^{-8}i; 1 - i; 1 + i; -2 - 3.745^{-9}i]$
P_6	$[8; 5; 6; 7]$	209	$[0.0608; 0.0291; 0.0507; 0.0392]$
	$[0.01; 0.02; -0.03; 0.07]$	13	$[0.0466; 0.0618; 0.0281; 0.0433]$
	$[10; 1000; 10^6; 10^8]$	719	$[0.0447; 0.0610; 0.0289; 0.0452]$
P_7	$[4; 5; 6; 7]$	64	$[3.0007; 2.0007; 1.9992; 2.9992]$
	$[2.1; 2.5; 2.9; 3.3]$	10	$[1.9993; 2.0006; 2.9996; 3.0003]$

Table 1: Resultados do algoritmo para alguns polinômios

Na Tabela 1 a primeira coluna diz respeito ao polinômio testado, a segunda ao valor inicial. A terceira ao número de iterações, e a última ao vetor encontrado pelo método.

DOI: 10.5540/03.2014.002.01.0063

4 Conclusão

O fato de o algoritmo gastar maior número de iterações para valores iniciais mais distantes da raiz, é característico do método de Newton empregado. Se p tiver raiz nula ou complexa, o método mostrou funcionar adequadamente. No entanto, não mostrou ser adequado quando o valor inicial tem componentes repetidas. Isto, porque a matriz de Vandermonde gerada ficará singular, resultando num sistema sem solução, o que implica num problema de estabilidade quando se lida com polinômios de raízes de multiplicidade maior que 1. Esta pesquisa ainda está em andamento, de modo que buscamos métodos melhores para se aplicar a resolução do sistema de Vandermonde. Os testes com o método de newton puro apenas servem para ilustrar como o sistema de Girard transformado pode ser útil no cálculo de raízes de polinômios.

References

[1] P. Barry, *Triangle Geometry and Jacobsthal Numbers*, "Irish Mathematical Society Bulleti", 51 (2003), p. 45-57.

[2] B. C. Berndt and W. Chu, *Two Entries on Bilateral Hypergeometric Series in Ramanujan's Lost Notebook*, "Proceedings of American Mathematical Society", 135-1 (2007), p. 129-134.

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations, third edition*, "The Johns Hopkins University Press", 1996.

[4] M. E. Horn, *Pascal Pyramids, Pascal Hyper-Pyramids and Bilateral Multinomial Theorem*, Am Neuen Palais 10, D - 14469 Potsdam, Germany.

[5] M. E. Horn, *Bilateral Binomial Theorem*, "Siam-Problem No. 001-03" , Cornell e-print-archive [2006-11-28]: www.siam.org/journals/categories/03-001.php

[6] B. Kallós, *The generalization of Pascal's triangle from algebraic point of view*, "Acta Academiae Paedagogicae Agriensis, Sectio Mathematicae", 24 (1997), p. 11-18.

[7] B. Kallós, *A Generalization of Pascal's triangle using powers of base numbers*, "Annales Mathématiques", 13-1 (2006), P. 1-15

DOI: 10.5540/03.2014.002.01.0063

[8] F. Nour-Eddine, *Polynomial Triangle Revisited*, Cornell University Library, Mathematics-Combinatorics 2012arXiv1202.0228F, 2012.

[9] J. Nocedal and S.J. Wright, *Numerical Optimization*, "Springer Series in Operations Research", New York, 1999.

DOI: 10.5540/03.2014.002.01.0063