Journal of Computational and Applied Mathematics 272 (2014) 276–292

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Solving secular and polynomial equations: A multiprecision algorithm

Dario A. Bini [a,*], Leonardo Robol [b]

[a] Dipartimento di Matematica, Università di Pisa, Italy

[b] Scuola Normale Superiore, Pisa, Italy

ARTICLE INFO

Article history:
Received 21 December 2012
Received in revised form 20 March 2013

Dedicated to Prof. Dario A. Bini's mother

Keywords:
Secular equations
Polynomial roots
Multiprecision computations
Ehrlich–Aberth iteration
Root neighborhoods

ABSTRACT

We present an algorithm for the solution of polynomial equations and secular equations of the form $S(x) = 0$ for $S(x) = \sum_{i=1}^{n} \frac{a_i}{x-b_i} - 1 = 0$, which provides guaranteed approximation of the roots with any desired number of digits. It relies on the combination of two different strategies for dealing with the precision of the floating point computation: the strategy used in the package MPSolve of D. Bini and G. Fiorentino [D.A. Bini, G. Fiorentino, Design, analysis and implementation of a multi-precision polynomial rootfinder, Numer. Algorithms 23 (2000) 127–173] and the strategy used in the package Eigensolve of S. Fortune [S. Fortune, An iterated eigenvalue algorithm for approximating the roots of univariate polynomials, J. Symbolic Comput. 33 (5) (2002) 627–646]. The algorithm is based on the Ehrlich–Aberth (EA) iteration, and on several results introduced in the paper. In particular, we extend the concept and the properties of root-neighborhoods from polynomials to secular functions, provide perturbation results of the roots, obtain an effective stop condition for the EA iteration and guaranteed *a posteriori* error bounds. We provide an implementation, released in the package MPSolve 3.0, based on the GMP library. From the many numerical experiments it turns out that our code is generally much faster than MPSolve 2.0 and of the package Eigensolve. For certain polynomials, like the Mandelbrot or the partition polynomials the acceleration is dramatic. The algorithm exploits the parallel architecture of the computing platform.

1. Introduction

Given a positive integer n and complex numbers a_i, b_i, $i = 1, \ldots, n$ such that $b_i \neq b_j$ for $i \neq j$ and $a_i \neq 0$ for $i = 1, \ldots, n$, consider the rational function

$$S(x) = \sum_{i=1}^{n} \frac{a_i}{x - b_i} - 1. \tag{1}$$

We associate with $S(x)$ the *secular equation* $S(x) = 0$. The numbers b_i and a_i, for $i = 1, \ldots, n$ are referred to as the nodes and the coefficients of the secular function $S(x)$, respectively.

Equations of this kind are mainly encountered in the case of real nodes b_i and positive coefficients a_i when the roots are real. Typical examples are modifying symmetric eigenvalue problems [1], or solving the tridiagonal symmetric eigenvalue problem by means of divide and conquer techniques [2,3], updating the singular values of a matrix, solving least squares

* Corresponding author. Tel.: +39 0502213279.
E-mail addresses: bini@dm.unipi.it, dario.a.bini@gmail.com (D.A. Bini), leonardo.robol@sns.it (L. Robol).

0377-0427/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cam.2013.04.037

problems [4], invariant subspace computation [5] and more. Over the complex field, for any set of nodes and coefficients, secular equations are encountered in the solution of the eigenvalue problem for a diagonal plus rank-one matrix [6–9] and in representing generalized companion matrix pencils in the Lagrange basis especially in the framework of "polynomial algebra by values" [10].

Secular equations are also a powerful tool for attacking the polynomial root-finding problem. This is the main fact that motivates our interest in such equations. In fact, the monic polynomial of degree n

$$p(x) = \Pi(x)S(x), \qquad \Pi(x) = -\prod_{i=1}^{n}(x - b_i)$$

has roots that coincide with the roots of $S(x)$. Moreover, one can verify that

$$a_i = \frac{p(b_i)}{\prod_{j=1,\, j\neq i}^{n}(x - b_j)} \tag{2}$$

so that, given the polynomial $p(x)$ and the nodes b_i it is not expensive to compute the coefficients a_i and to reformulate the polynomial root-finding problem in terms of a secular equation.

In this paper we present a method for the numerical solution of secular equations together with its computational analysis. More precisely we describe, analyze and implement an algorithm which, given in input the coefficients a_i and the nodes b_i, $i = 1, \ldots, n$ of the secular function $S(x)$ together with an integer d, provides in output the roots of $S(x)$ represented with d guaranteed digits. In its cheaper version (isolation), the algorithm can provide approximations to the roots with the minimum number of digits sufficient to separate them from each other. The maximum number d of digits is used only for those roots, if any, which cannot be otherwise separated.

The algorithm can be effectively used as a tool for computing an arbitrarily large number d of digits of the roots of a polynomial $p(x)$ assigned either in terms of its coefficients in some polynomial basis or by means of a black box which, given in input a complex number x, provides in output the complex number $p(x)$. In fact we will show that using the representation of a polynomial in terms of secular equation provides substantial computational advantages.

The method relies on the combination of two different strategies to reduce the required precision of the floating point arithmetic: the strategy adopted in the package of MPSolve of D. Bini and G. Fiorentino [11], and that used in the package Eigensolve of S. Fortune [12]. It exploits some theoretical results, that we present in this paper, concerning root-neighborhoods, numerical conditioning, *a posteriori* error bounds, and rounding error analysis, related to computations with secular functions. It relies also on the formulation of the problem given in terms of structured matrices and on the Ehrlich–Aberth iteration as main approximation engine [13,14].

The algorithm that we have obtained has been implemented in the language C and incorporated in the package MPSolve originating the release MPSolve 3.0. The software is free and can be downloaded from http://riccati.dm.unipi.it/mpsolve. It enables to deal with secular and polynomial equations where the real or complex input data can take either the approximate form of floating point numbers or the exact form of integers and rationals. The implementation exploits the parallel architecture of the computing platform.

From the many numerical experiments that we have performed our code, even though applied without the parallelism, is generally faster than MPSolve and Eigensolve. For certain polynomials it is dramatically faster. The speed up that it reaches when using multicore hardware is close to optimal. Just to make an example, for the partition polynomial of degree 72.000 which has integer coefficients representable with several megabytes, MPSolve 2.0 took about 30 days to compute all the roots [15]. Our code computes the roots in less than 2 h whereas Eigensolve has an estimated CPU time of many years.

The paper is organized as follows. In Section 2 we recall the strategies of MPSolve and Eigensolve, and give an overview of our algorithm. In Section 3 we develop the numerical tools that we need. In particular we provide a backward stable method for computing $S(x)$, and we extend the definition and the properties of root-neighborhoods [16,17] from polynomials to secular functions. These properties allow us to devise effective stop conditions to halt the Ehrlich–Aberth iteration. Section 4 deals with the matrix representation of the problem and with the way of constructing different secular functions having the same roots as $S(x)$ and using different sets of nodes. We refer to these functions as *equivalent functions*. Gerschgorin-like inclusion results are given and used for devising *a posteriori* error bounds. In Section 5 we perform a perturbation analysis of the roots of secular functions where we show that the condition number of the roots converges to zero as the nodes, used for representing the equivalent secular function, converge to the roots. The Ehrlich–Aberth iteration is recalled in Section 6. Finally, in Section 7 we present the results of the numerical experiments.

2. Overview of the algorithm

Our algorithm performs computations in floating point arithmetic with a variable number of digits. Since high precision arithmetic is expensive, our goal is to keep the number of digits of the floating point computation as low as possible. We will refer to the *working precision* as to the number w of binary digits used in the current floating point computation and denote $u = 2^{-w}$ the corresponding *machine precision*. Recall that the standard IEEE double precision arithmetic has $w = 53$ bits.