Symbolic computation of the roots of nonlinear algebraic equations using perturbation theory

Gürhan Ustalı\(^1\), Ünal Göktas\(^2\), Mehmet Pakdemirli\(^3\)

\(^1\)Department of Computer Programming, Fatih University, Yenimahalle, Ankara 06370, Turkey
gustali@fatih.edu.tr
\(^2\)Department of Computer Engineering, Turgut Özal University, Keçiören, Ankara 06010, Turkey
ugoktas@turgutozal.edu.tr
\(^3\)Department of Mechanical Engineering, Celal Bayar University, Muradiye, Manisa 45140, Turkey
mpak@bayar.edu.tr

Abstract—Two algorithms, one for estimating the magnitudes of the roots of a polynomial equation before actually solving it, and one for computing the roots of nonlinear algebraic equations using perturbation theory, are presented. The algorithms are illustrated on an example. The Mathematica package PerturbationIterationFindRootAlgorithm.m, carries out all the steps of these algorithms automatically. The package is briefly discussed.

Keywords—perturbation theory, nonlinear equations

I. INTRODUCTION

As many real life problems are modeled by nonlinear algebraic equations which display the interrelations between mathematics, applied sciences and engineering, finding roots of nonlinear algebraic equations has always been a problem of interest [1]-[4].

Many methods have been developed to compute the roots of nonlinear algebraic equations such as Newton-Raphson, Muller’s, Secant, and Householder’s iteration methods [2]-[5].

Perturbation theory is widely used to compute approximate solutions of algebraic equations, differential equations, difference equations, etc. [3], [4], [6]-[8].

In the book by Nayfeh [6], many examples of algebraic equations with small parameters are treated using perturbation methods. In perturbation methods, the root is first expanded in a perturbation series, and then each correction term is calculated once to find the approximate root. However, no iterations over the corrections are carried. Perturbation and iteration methods are treated separately in the book by Hinch [7].

In this study, we will be using the so-called perturbation-iteration method, a combination of perturbations with iterations [2]-[4]. Perturbation-iteration method, depending on the number of terms in the perturbation series expansion \((n)\), the number of terms in the Taylor series expansion \((m)\), and the way the resulting equations are separated, gives rise to the well-known iteration formulas such as Newton-Raphson and Householder’s iteration, along with many other iteration formulas [2]-[4].

In iterative algorithms, a good initial estimate is essential in order to converge to a root. When working with polynomials, Theorems 1 and 2 in [1], based on the order of magnitude concept of the perturbation theory, can be used for a good initial estimate.

We have developed a Mathematica package PerturbationIterationFindRootAlgorithm.m, for general \(n\) and \(m\) such that \(\leq n \leq m\), which carries out all the steps of the perturbation-iteration algorithms automatically, including the computation of an initial guess for the root when dealing with polynomials.

The paper is organized as follows. Section 2 outlines the details of the initial root estimation for polynomials. The details of the perturbation-iteration algorithm for finding the approximate solutions of the nonlinear algebraic equations are given in Section 3. In Section 4 comparison of our results with earlier results is presented. Our Mathematica package PerturbationIterationFindRootAlgorithm.m is briefly discussed in Section 5.

II. INITIAL ROOT ESTIMATION FOR POLYNOMIALS

Two theorems are given in [1] for estimating the magnitudes of the roots of polynomials.

The first theorem tells that, for the polynomial equation

\[ a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0, \]  

if all the coefficients \( a_i \) \((i = 0, 1, \ldots, n)\) are of the same order of magnitude, then the root is of \(O(1)\).

The second theorem says that, for the polynomial equation

\[ a_n x^n + a_{n-1} x^{n-2} + \cdots + a_1 x + a_0 = 0, \]  

if \( a_m \sim O(1/\varepsilon^k) \) \((k > 0)\) with all other coefficients of being \(O(1)\), then the possible roots are of either \(O(\varepsilon^{k/m})\) \((for m \neq 0)\) or of \(O(1/\varepsilon^{k/(n-m)})\) \((for m \neq n)\).

Example: Let us consider the equation \(x^3 + 0.5x^2 - x + 1200 = 0\). Although the theorems are universal and valid for any \(\varepsilon\) as long as \(\varepsilon\) is much smaller than 1, we select \(\varepsilon = 0.1\) for our computations. So, we have \(a_0 = 1200 \sim O(1/\varepsilon^2)\). Hence \(k = 3, m = 0\), and \(n = 3\). Since \(m = 0\), the root is of \(O(1/\varepsilon^{3/(3-0)}) = O(1/\varepsilon) = O(10)\), and hence \(|x| \approx 10\) according to the second theorem. Note that, the numerically calculated roots \(-10.8278\) and \(5.1639 \pm 9.1739 i\) are all approximately 10 in magnitude [1].
III. PERTURBATION-ITERATION METHOD

Depending on the number of terms in the perturbation expansion, number of terms in the Taylor series expansion, and the way the resulting equations are grouped, different iteration formulas come out [2]. For simplicity, we will show the details of \( n = 1, m = 1 \) (so-called single correction-term) case.

Let us consider the nonlinear equation
\[
f(x) = 0.
\]
To find the roots of (3), add a single correction term to the initial assumed root:
\[
x = x_0 + \varepsilon x_1.
\]
Inserting (4) into (3) and expanding in a Taylor series yields
\[
f(x_0) + f'(x_0) \varepsilon x_1 = 0.
\]
Solving (5) for \( \varepsilon x_1 \) and inserting into (4), we get
\[
x = x_0 - f(x_0)/f'(x_0).
\]
This becomes the Newton-Raphson method when put into an iteration form as:
\[
x_{n+1} = x_n - f(x_n)/f'(x_n).
\]

IV. NUMERICAL COMPARISONS

Comparisons of different perturbation-iteration algorithms are given in Tables I-III. As the number of correction terms increases, the algorithms behave slightly better by converging to the numerical result with less iterations on these three test equations. Our results match with the results in [2].

<table>
<thead>
<tr>
<th>Table I</th>
<th>NUMBER OF ITERATIONS FOR FINDING THE ROOTS OF ( e^{-x} - x = 0 ) BY DIFFERENT PERTURBATION-ITERATION ALGORITHMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>Newton-Raphson ((n = 1, m = 1))</td>
</tr>
<tr>
<td>Initial value</td>
<td>0.0</td>
</tr>
<tr>
<td>First Iteration</td>
<td>0.5</td>
</tr>
<tr>
<td>Second Iteration</td>
<td>0.566311</td>
</tr>
<tr>
<td>Third Iteration</td>
<td>0.567143</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table II</th>
<th>NUMBER OF ITERATIONS FOR FINDING THE ROOTS OF ( \tan(x) - \tan(x) = 0 ) BY DIFFERENT PERTURBATION-ITERATION ALGORITHMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>Newton-Raphson ((n = 1, m = 1))</td>
</tr>
<tr>
<td>Initial value</td>
<td>4.0</td>
</tr>
<tr>
<td>First Iteration</td>
<td>3.92225</td>
</tr>
<tr>
<td>Second Iteration</td>
<td>3.9266</td>
</tr>
</tbody>
</table>

V. THE MATHEMATICA PACKAGE

Two subfunctions are of importance in our Mathematica package, EstimateRootsOfPolynomials and PerturbationIterationFindRootAlgorithm. The first is for estimating an initial value for the root when working with polynomials, and the latter is for computing the root of a nonlinear algebraic equation starting with an initial guess. To use the code, first load the Mathematica package PerturbationIterationFindRootAlgorithm.m. [9] using the command:
\[
\text{In}[2]:= \text{Get}[	ext{"PerturbationIterationFindRootAlgorithm.m"}];
\]
In order to get an initial estimate for the root of \( x^3 - 2x^2 - x = -50 \), just type in:
\[
\text{In}[3]:= \text{EstimateRootOfPolynomials}[\{x^3 - 2x^2 - x = -50 \}, x ]
\]
and the result will come out as:
\[
\text{Out}[3]:= \{2.15443\}
\]
In order to find the roots of the polynomial equation \( x^3 - 3x = 50 \), with \( n = 1, m = 2 \) type in:
\[
\text{In}[4]:= \text{PerturbationIterationFindRootAlgorithm}[\{x^3 - 3x = 50 \}, x, 1, 2 ]
\]
and the two roots will come out as:
\[
\text{Out}[4]:= \{-5.72842, 8.72842\}
\]
Note that in the polynomial case estimation routine is called automatically. For the non-polynomial case, the initial guess for the root has to be given as an extra argument. Details can be found in [9].
VI. CONCLUSION

A Mathematica package for carrying out the tedious calculations needed to find the roots of nonlinear algebraic equations using perturbation-iteration methods has been developed [9].

The package is general enough to work with any $n$ and $m$ such that $1 \leq n \leq m$.

Preliminary test results suggest that as the number of correction terms increase, less iterations are needed for convergence to the root.

REFERENCES