Available online at www.sciencedirect.com
ScienceDirect
Computers and Mathematics with Applications 56 (2008) 1114–1120
An International Journal computers & mathematics with applications
www.elsevier.com/locate/camwa

Accurate simple zeros of polynomials in floating point arithmetic

Stef Graillat

UPMC Univ Paris 06, CNRS, UMR 7606, LIP6, 4 place Jussieu, F-75252, Paris cedex 05, France

Received 26 December 2007; accepted 6 February 2008

Abstract

In the paper, we examine the local behavior of Newton's method in floating point arithmetic for the computation of a simple zero of a polynomial assuming that an good initial approximation is available. We allow an extended precision (twice the working precision) in the computation of the residual. We prove that, for a sufficient number of iterations, the zero is as accurate as if computed in twice the working precision. We provide numerical experiments confirming this.

Keywords: Zeros of polynomials; Newton's method; Condition number; Floating point arithmetic

1. Introduction and notation

The key to computing an accurate solution to a nonlinear equation is the accurate evaluation of the function in use. In this paper, our purpose is to compute accurate simple zeros of univariate polynomials relying on Newton's method and assuming that an good initial approximation is available. To reach this goal, we focus on two important things:

- explaining what we mean by "accurate solution";
- having an accurate polynomial evaluation algorithm to compute the residual in the Newton's iteration.

Let us explain now what we mean by "accurate solution". Let $\widehat{x}$ be the computed solution of a problem (P) whose exact solution is x. Suppose that the computations have been done with a t-bit floating point arithmetic. We will say the $\widehat{x}$ is as accurate as if computed with twice the working precision if

$$\frac{|\widehat{x}-x|}{|x|} \leq \mathtt{eps} + C\mathtt{eps}^2\mathrm{cond}(P), \tag{1}$$

where C is a moderate constant, $\mathtt{eps} = 2^{-t}$, $|\cdot|$ is a norm on the space of the solution and $\mathrm{cond}(P)$ is the condition number of the problem (P). In the right-hand side of inequality (1), the second term reflects the computation in twice the working precision and the first one the rounding into the working precision. Relation (1) is what we called the compensated rule of thumb, the classic rule of thumb being [1, p. 9]

$$\frac{|\widehat{x}-x|}{|x|} \leq C\mathtt{eps}\,\mathrm{cond}(P).$$

E-mail address: stef.graillat@lip6.fr.

0898-1221/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2008.02.027

Throughout the paper, we assume working with a floating point arithmetic adhering to IEEE 754 floating point standard [2]. We assume that neither overflow nor underflow occur. The set of floating point numbers is denoted by $\mathbf{F}$ and the relative rounding error by $\mathtt{eps}$. For IEEE 754 double precision we have $\mathtt{eps} = 2^{-53}$ and for IEEE 754 single precision $\mathtt{eps} = 2^{-24}$.

We denote by $\mathrm{fl}(\cdot)$ the result of a floating point computation, where all operations inside parentheses are done in floating point working precision. Floating point operations in IEEE 754 satisfy [1]

$$\mathrm{fl}(a \circ b) = (a \circ b)(1 + \varepsilon) = \quad \text{for } \circ = \{+, -, \cdot, /\} \text{ and } |\varepsilon| \leq \mathtt{eps}.$$

This implies that

$$|a \circ b - \mathrm{fl}(a \circ b)| \leq \mathtt{eps}|a \circ b| \text{ and } |a \circ b - \mathrm{fl}(a \circ b)| \leq \mathtt{eps}|\mathrm{fl}(a \circ b)| \text{ for } \circ = \{+, -, \cdot, /\}.$$

We use standard notation for error estimations. The quantities γ_n are defined as usual [1] by

$$\gamma_n := \frac{n\mathtt{eps}}{1 - n\mathtt{eps}} \quad \text{for } n \in \mathbf{N},$$

where we implicitly assume that $n\mathtt{eps} \leq 1$.

The rest of the paper is organized as follows. In Section 2, we recall some results on the Horner scheme, error-free transformations and the Compensated Horner scheme. In Section 3, we present the condition number of a simple zero. In Section 4, we present Newton's method for root-finding using the Compensated Horner scheme to compute the residual. In Section 5, we give some numerical experiments. Finally, we conclude by giving some hints about future work.

2. Accurate polynomial evaluation

In this section, we first recall the Horner scheme as well as give an error bound. We then recall the classic error-free transformations. We use these transformations for a Compensated Horner scheme which gives a result as accurate as if computed by the classic Horner scheme using twice the working precision and then rounded to the working precision.

2.1. Classic Horner scheme

The classic method for evaluating a polynomial

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

is the Horner scheme which consists in the following algorithm.

Algorithm 1. Polynomial evaluation with Horner's scheme

function $\mathtt{res} = \mathtt{Horner}(p, x)$
  $s_n = a_n$
  for $i = n - 1 : -1 : 0$
    $s_i = s_{i+1} \cdot x + a_i$
  end
  $\mathtt{res} = s_0$

A forward error bound is (see [1, p. 95]):

$$|p(x) - \mathtt{Horner}(p, x)| \leq \gamma_{2n} \sum_{i=0}^{n} |a_i||x|^i = \gamma_{2n} \widetilde{p}(|x|), \tag{2}$$

where $\widetilde{p}(x) = \sum_{i=0}^{n} |a_i| x^i$.

2.2. Error-free transformations (EFT)

One can notice that $a \circ b \in \mathbf{R}$ and $\mathrm{fl}(a \circ b) \in \mathbf{F}$ but in general we do not have $a \circ b \in \mathbf{F}$. It is known that for the basic operations $+, -, \cdot$, the approximation error of a floating point operation is still a floating point number (see for example [3]):

$$\begin{aligned} &x = \mathrm{fl}(a \pm b) \Rightarrow a \pm b = x + y \quad \text{with } y \in \mathbf{F}, \\ &x = \mathrm{fl}(a \cdot b) \Rightarrow a \cdot b = x + y \quad \text{with } y \in \mathbf{F}, \end{aligned} \tag{3}$$

where no underflow is assumed for multiplication. These are *error-free* transformations of the pair (a, b) into the pair (x, y).

Fortunately, the quantities x and y in (3) can be computed exactly in floating point arithmetic by applying the well known algorithms for error-free summation and multiplication, namely Knuth's `TwoSum` from [4, Thm B. p. 236] and Dekker–Veltkamp's `TwoProduct` from [3].

The following theorem summarizes the properties of algorithms `TwoSum` and `TwoProduct`.

Theorem 1 (*Ogita, Rump and Oishi [5]*). *Let $a, b \in \mathbf{F}$ and let $x, y \in \mathbf{F}$ such that $[x, y] = \mathtt{TwoSum}(a, b)$. Then,*

$$a + b = x + y, \quad x = \mathrm{fl}(a + b), \quad |y| \leq \mathtt{eps}|x|, \quad |y| \leq \mathtt{eps}|a + b|. \tag{4}$$

The algorithm `TwoSum` *requires* 6 *flops.*

Let $a, b \in \mathbf{F}$ and let $x, y \in \mathbf{F}$ such that $[x, y] = \mathtt{TwoProduct}(a, b)$. Then,

$$a \cdot b = x + y, \quad x = \mathrm{fl}(a \cdot b), \quad |y| \leq \mathtt{eps}|x|, \quad |y| \leq \mathtt{eps}|a \cdot b|. \tag{5}$$

The algorithm `TwoProduct` *requires* 17 *flops.*

We present now an error-free transformation for the polynomial evaluation with Horner scheme.

Algorithm 2 (*Graillat, Langlois and Louvet [6]*). Error-free transformation for the Horner scheme

function $[h, p_\pi, p_\sigma] = \mathtt{EFTHorner}(p, x)$
  $s_n = a_n$
  for $i = n - 1 : -1 : 0$
    $[p_i, \pi_i] = \mathtt{TwoProduct}(s_{i+1}, x)$
    $[s_i, \sigma_i] = \mathtt{TwoSum}(p_i, a_i)$
    Let π_i be the coefficient of degree i in p_π
    Let σ_i be the coefficient of degree i in p_σ
  end
  $h = s_0$

The next theorem proves that Algorithm 2 is an error-free transformation.

Theorem 2 (*Graillat, Langlois and Louvet [6]*). *Let $p(x) = \sum_{i=0}^{n} a_i x^i$ be a polynomial of degree n with floating point coefficients, and let x be a floating point value. Let $[h, p_\pi, p_\sigma] = \mathtt{EFTHorner}(p, x)$ (Algorithm 2). Then*

(i) *the floating point evaluation $h = \mathtt{Horner}(p, x)$ and*
(ii) *two polynomials p_π and p_σ of degree $n - 1$ with floating point coefficients,*

satisfies

$$p(x) = h + (p_\pi + p_\sigma)(x). \tag{6}$$

Algorithm 2 requires $23n$ *flops.*

2.3. Compensated Horner scheme

From Theorem 2, the global forward error affecting the floating point evaluation of p at x according to the Horner scheme is

$$e(x) = p(x) - \texttt{Horner}(p, x) = (p_\pi + p_\sigma)(x).$$

The coefficients of these polynomials are exactly computed by Algorithm 2, together with $\texttt{Horner}(p, x)$. Indeed, if $[h, p_\pi, p_\sigma] = \texttt{EFTHorner}(p, x)$, then p_π and p_σ are two exactly representable polynomials. The key to increase the accuracy of the computed result is to compute an approximate of the global error $e(x)$ in working precision, and then to compute a corrected result

$$\texttt{res} = \text{fl}(\texttt{Horner}(p, x) + e(x)).$$

We say that $c = \text{fl}(e(x))$ is a corrective term for $\texttt{Horner}(p, x)$. The corrected result res is expected to be more accurate than the first result $\texttt{Horner}(p, x)$.

Our aim is now to compute the corrective term $c = \text{fl}((p_\pi + p_\sigma)(x))$. For that we evaluate the polynomial whose coefficients are those of $p_\pi + p_\sigma$ rounded to the nearest floating point value. This process is described by Algorithm 3.

Algorithm 3. Evaluation of the sum of two polynomials.

function $\texttt{res} = \texttt{HornerSum}(p, q, x)$
  $r_n = \text{fl}(a_n + b_n)$
  for $i = n - 1 : -1 : 0$
    $r_i = \text{fl}(r_{i+1} \cdot x + (a_i + b_i))$
  end

We can now describe the Compensated Horner Scheme.

Algorithm 4 (*Graillat, Langlois and Louvet [6]*). Compensated Horner scheme

function $\texttt{res} = \texttt{CompHorner}(p, x)$
  $[h, p_\pi, p_\sigma] = \texttt{EFTHorner}(p, x)$
  $c = \texttt{HornerSum}(p_\pi, p_\sigma, x)$
  $\texttt{res} = \text{fl}(h + c)$

The following theorem proves that the result of a polynomial evaluation computed with the Compensated Horner scheme (Algorithm 4) is as accurate as if computed by the classic Horner scheme using twice the working precision and then rounded to the working precision.

Theorem 3 (*Graillat, Langlois and Louvet [6]*). *Given a polynomial $p = \sum_{i=0}^{n} a_i x^i$ of degree n with floating point coefficients, and x a floating point value. We consider the result $\texttt{CompHorner}(p, x)$ computed by Algorithm 4. Then,*

$$|\texttt{CompHorner}(p, x) - p(x)| \leq \texttt{eps}|p(x)| + \gamma_{2n}^2 \widetilde{p}(x). \tag{7}$$

The Algorithm CompHorner *requires $26n + 3$ flops.*

3. Condition number for root finding

Given a problem, we want to know how to measure the difficulty of solving it. This will be done via the notion of *condition number*. Roughly speaking, the condition number measures the sensitivity of the solution to perturbation in the data. Here is the classic definition for the condition number of root finding for simple roots.

Definition 4. Let $p(z) = \sum_{i=0}^{n} a_i z^i$ be a polynomial of degree n and x be a simple zero of p. The *condition number* of x is defined by

$$\text{cond}(p, x) = \lim_{\varepsilon \to 0} \sup \left\{ \frac{|\Delta x|}{\varepsilon |x|} : |\Delta a_i| \leq \varepsilon |a_i| \right\}.$$

In the previous definition, Δx represents the variation of the zero x when the polynomial is perturbed by a polynomial $\Delta p(z) = \sum_{i=0}^{n} \Delta a_i z^i$. It means that $x + \Delta x$ is a zero of $p + \Delta p$.

The following theorem gives an explicit formula to compute the condition number.

Theorem 5 (*Chaitin–Chatelin and Frayssé [7]*). *Let p be a polynomial of degree n and x be a simple zero of p. The condition number of x is given by*

$$\operatorname{cond}(p, x) = \frac{\widetilde{p}(|x|)}{|x||p'(x)|}.$$

4. Accurate Newton's method

In this section, we specialize the result from [8] (see also [9]) for the Newton's method in floating point arithmetic in the case of an univariate polynomial with a simple root. We use the Compensated Horner scheme to accurately compute the residual. In that case, we show that the computed result (an approximation of the simple root of the polynomial) is as accurate as if computed with twice the working precision via the classic Newton's method and then rounded back to the working precision.

In [8], Tisseur provided a comprehensive analysis of the Newton's method in floating point arithmetic (see also [1, chap. 25]) for solving the equation $F(x) = 0$ where $F : \mathbf{R}^m \to \mathbf{R}^m$ is continuously differentiable and J the Jacobian matrix $(\partial F_i / \partial x_j)$ of F is Lipschitz continuous.

Let p be a given polynomial with simple zeros. We apply the Newton's method with $F(x) = p(x)$ and so $J(x) = p'(x)$. The classic Newton's method is Algorithm 5.

Algorithm 5. Classic Newton's method

$$x_0 = \xi$$
$$x_{i+1} = x_i - \frac{p(x_i)}{p'(x_i)}$$

Hereafter, we use the compensated Horner scheme to evaluate the residual $p(x)$ in order to get a result as accurate as if computed in twice the working precision. We also assume that we already know that the root we are looking for belongs to $[a, b]$ with $a, b \in \mathbf{R}$. We also define $\beta = \max_{x \in [a,b]} |p'(x)|$. The accurate Newton's method is Algorithm 6.

Algorithm 6. Accurate Newton's method

$$x_0 = \xi$$
$$x_{i+1} = x_i - \frac{\texttt{CompHorner}(p, x_i)}{p'(x_i)}$$

In Algorithm 6, we use `CompHorner` to evaluate the residual $p(x_i)$ but not $p'(x_i)$. Indeed, since x is a simple zero, x is not a zero of p'. Therefore, the evaluation of p' near x is not ill-conditioned, and so Horner's classical scheme is suitable.

Applying Corollary 2.3 of [8] with $F(x) = p(x)$ and $J(x) = p'(x)$, we obtain the following theorem.

Theorem 6. *Assume that there is an x such that $p(x) = 0$ and $p'(x) \neq 0$ is not too small. Assume also that*

$$\texttt{eps} \cdot \operatorname{cond}(p, x) \leq 1/8 \quad \textit{for all } i.$$

Then, for all x_0 such that

$$\beta |p'(x)^{-1}| |x_0 - x| \leq 1/8,$$

Newton's method in floating point arithmetic generates a sequence of $\{x_i\}$ whose relative error decreases until the first i for which

$$\frac{|x_{i+1} - x|}{|x|} \approx \texttt{eps} + \gamma_{2n}^2 \operatorname{cond}(p, x). \tag{8}$$

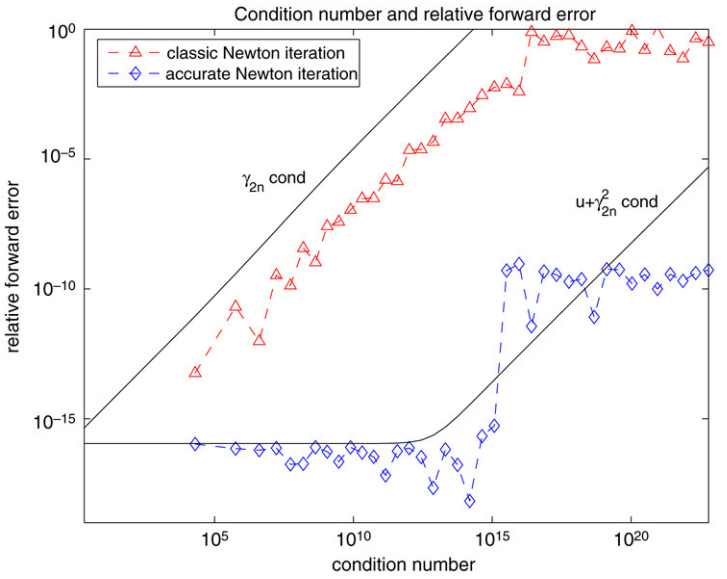

Fig. 1. Accuracy of the classic Newton iteration and of the accurate Newton iteration.

The theorem means that if we begin not too far from the simple root, the Newton's method gives an approximation of the root as accurate as if computed with twice the working precision.

The use of an accurate polynomial evaluation algorithm is essential. Indeed, if we use the classic Horner scheme, then, at the end of the iteration, we only have

$$\frac{|x_{i+1} - x|}{|x|} \approx \gamma_{2n}\,\mathrm{cond}(p, x). \tag{9}$$

5. Numerical experiments

All our experiments are performed using the IEEE 754 double precision with MATLAB 7. When needed, we use the Symbolic Math Toolbox to accurately compute the roots of polynomials (in order to compute the relative forward error).

We test the Newton's iterations on the expanded form of the polynomial $p_n(x) = (x-1)^n - 10^{-8}$ for $n = 1{:}40$. The condition number $\mathrm{cond}(p_n, x)$ where x is the root $1 + 10^{-8/n}$ varies from 10^4 to 10^{22}.

Fig. 1 shows the relative accuracy $|\widehat{x} - x|/|x|$ where x is the exact root and $\widehat{x}$ is the computed value by the two Algorithms 5 and 6. We also plot the *a priori* error estimation (8) and (9).

As we can see in Fig. 1, the accurate Newton's iteration exhibits the expected behavior, that is to say, the compensated rule of thumb. As long as the condition number is less that 10^{15}, the accurate Newton's iteration produces results with full precision (forward relative error of the order of 10^{-16}). For condition numbers greater than 10^{15}, the accuracy decreases.

6. Conclusion and future work

In the paper, we have proved that the Newton's iteration makes it possible to refine a close initial approximation of a simple root to yield an approximation as accurate as if computed with twice the working precision.

We only dealt with simple roots. If the root has multiplicity $m > 1$, one can use the modified Newton's iteration as follows.

Algorithm 7. Modified Newton's method

$x_0 = \xi$

$x_{i+1} = x_i - m\frac{p(x_i)}{p'(x_i)}$

A future work will be to see if we can get the same kind of results than for simple roots when we already know the multiplicity of the root.

Finally, we will study also the modification where Newton's method is applied to the ratio $p(x)/p'(x)$ to approximate multiple roots of $p(x)$ having an unknown multiplicity.

Acknowledgment

I am very grateful to Professor Victor Pan for his valuable comments and suggestions.

References

[1] N.J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd ed, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.
[2] IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985, Institute of Electrical and Electronics Engineers, New York, 1985. Reprinted in SIGPLAN Notices, 22(2) (1987) 9–25.
[3] T.J. Dekker, A floating-point technique for extending the available precision, Numer. Math. 18 (1971) 224–242.
[4] D.E. Knuth, The Art of Computer Programming, volume 2, Seminumerical Algorithms, 3rd ed, Addison-Wesley, Reading, MA, USA, 1998.
[5] T. Ogita, S.M. Rump, S. Oishi, Accurate sum and dot product, SIAM J. Sci. Comput. 26 (6) (2005) 1955–1988.
[6] S. Graillat, N. Louvet, P. Langlois, Compensated Horner scheme, Research report 04, Équipe de recherche DALI, Laboratoire LP2A, Université de Perpignan Via Domitia, France (July 2005).
[7] F. Chaitin-Chatelin, V. Frayssé, Lectures on finite precision computations, in: Software, Environments, and Tools, Society for Industrial and Applied Mathematics, SIAM, Philadelphia, PA, 1996.
[8] F. Tisseur, Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems, SIAM J. Matrix Anal. Appl. 22 (4) (2001) 1038–1057 (electronic).
[9] D.J. Bates, A.J. Sommese, J.D. Hauenstein, C.W. Wampler, Adaptive multiprecision path tracking, SIAM J. Numer. Anal. 46 (2) (2008) 722–746.