INTEGER ROOTS FOR INTEGER-POWER-CONTENT CALCULATIONS

D.J. JEFFREY, M.W. GIESBRECHT AND R.M. CORLESS

The Unive sity of Weste n Onta io, London, Onta io, Canada

E-mail: D.Jeff ey,M.Giesb echt,R.Co less@uwo.ca

In this extended abstract, we outline investigations of the application of Newton and Halley iteration to the computation of nth roots of an integer. We give an analysis that reduces the number of iterations by guaranteeing the number of correct digits obtained at each iteration. The initial application is to the calculation of the integer-power content of an integer.

1 Introduction

The *integer-power content* of $a \in \mathbb{Z}$ is defined[1] to be the largest $n \in \mathbb{Z}$ such that $a = b^n$ for some $b \in \mathbb{Z}$. We write $\mathrm{ipc}(a) = n$. Also, $b = \mathrm{ipf}(a)$ is the *integer-power free* base for a. Clearly we have $a = \mathrm{ipf}(a)^{\mathrm{ipc}(a)}$. Any algorithm that finds the integer-power content and base is called a perfect-power classification algorithm by Bernstein[2]. The first step in such an algorithm will be what Bernstein calls a perfect-power decomposition algorithm, that is the finding of any integer n, not necessarily maximal, such that $a = b^n$. Algorithms for computing n have been given by Bach & Sorenson[3], by Bernstein[2], and others. The computer alge ernst prtR Am by Bern aple g s a fu a drit is based on a progressive-precision Newton iteration. The use of progressively increasing precision at each step of an iteration is something that is obvious and instinctive to human computers working by hand, and was enunciated in a computer setting by Brent[4].

As well as examining the efficiency of existing algorithms, including p-adic methods, this paper considers in detail the control loops used in programming progressive-precision iteration. We also consider Halley iteration as an alternative to Newton iteration. Our considerations are also influenced by the specifics of the Maple computing environment, in particular the fact that Maple offers a radix 10 number system.

2 Newton and Halley Iteration

In this section, we give a uniform treatment of the Newton and Halley iterations.

2.1 General iteration formulae

Consider solving the equation $f(x) = 0$, given an initial estimate x_0 for the solution. We expand $f(x)$ as a Taylor series around x_0.

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \tfrac{1}{2}(x-x_0)^2 f''(x_0) + \tfrac{1}{6}(x-x_0)^3 f'''(x_0) + \ldots \quad (1)$$

Setting $h = x - x_0$ and $f(x) = 0$, we can solve for h by the Lagrange inversion theorem. Abbreviating $f(x_0)$ to f for clarity, we write

$$h = -\frac{1}{f'}f - \frac{f''}{2(f')^3}f^2 + \frac{3(f'')^2 - f'f'''}{6(f')^5}f^3 + \ldots \quad (2)$$

The series is written as shown to emphasize that it is a series in powers of $f(x_0)$. The classical Newton iteration is obtained by taking one term of this series, and the classical Halley iteration is obtained by converting the series to a continued fraction and taking terms to $O(f^2)$. The continued fraction form of (2) is

$$h = \cfrac{-f}{f' + \cfrac{-f}{2f'/f'' + \cfrac{-f}{3f'(f'')^2/(3(f'')^2 - 2f'f''')}}} \,, \quad (3)$$

and dropping higher-order terms and reverting to standard iteration notation, we get Halley's method as

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k) - \frac{1}{2}f(x_k)f''(x_k)/f'(x_k)} \,. \quad (4)$$