King [1] and Kiouselidis [2] have proposed a derivative free scheme that permits root finding methods, such as the secant method to preserve high order convergence when the root in question is multiple. In this note it is shown that the scheme can fail to achieve the maximum accuracy that is attainable at a fixed precision of computation.

This work was supported in part by the Office of Naval Research under contract No. N00014-76-C-0391.
The Behavior of a Multiplicity
Independent Root-finding
Scheme in the Presence of Error

G. W. Stewart

The problem considered in this note is that of finding a root of the equation

\[ f(x) = 0, \]

where \( f \) is a real valued function of a real variable. For simplicity we shall assume that \( f \) has a sufficient number of continuous derivatives for the purposes at hand.

There are any number of iterations, such as Newton's method or the secant method, that converge to a zero \( x^* \) of \( f \) from sufficiently near starting points. If \( x^* \) is a simple zero of \( f \), i.e. if \( f'(x^*) \neq 0 \), then many of these iterations converge superlinearly; however, if \( x^* \) is a multiple zero, the convergence is usually linear and can be quite slow. One way of circumventing this difficulty, an approach which goes back at least to Schröder [3], is to replace \( f \) by a function \( g \) that is constructed from \( f \) in such a way that \( x^* \) is a simple zero of \( g \). A classical choice is \( g = f/f' \). Recently King [1] and Kiouselidis [2] have proposed the function
(1) \[ g(x) = \frac{[f(x)]^2}{f[x + f(x)] - f(x)}, \]

where \( g \) and its derivatives are defined at \( x^* \) by continuity. It is easy to show that if \( x^* \) is a zero of multiplicity \( p \) of \( f \) then 
\[ g(x^*) = 0 \]
and
\[ g'(x^*) = \frac{1}{p}, \]

so that \( x^* \) is indeed a simple zero of \( g \).

This technique is appealing when it is used with a method like the secant method, which does not require derivatives; at the cost of one extra function evaluation per iteration it promises rapid convergence, free from considerations of multiplicity. However, it is important to realize that the technique has a hidden cost. When \( f \) is evaluated with error, an iteration based on it will not get as near the zero as a more conventional method. It is the purpose of this note to show why this is so.

The general situation is adequately illustrated by the function 
\[ f(x) = cx^p, \]
which has a zero of multiplicity \( p \) at zero. Suppose that in practice we cannot evaluate \( f(x) \) exactly but instead must work with a perturbed value.
\[ f(x) = f(x) + e(x), \]

where the error function \( e(x) \) satisfies

\[ |e(x)| \leq \varepsilon. \]

Now in the interval

\[ \left[ \frac{1}{c_p}, \frac{1}{c} \right] \]

the value of \( |f(x)| \) is not greater than \( \varepsilon \). Hence the value of \( \bar{f} \)
may be positive or negative at any point in the interval; i.e. any point
in (2) is a potential zero. Wilkinson [4] has shown that if \( e(x) \)
is noisy, as it will be when it is due to rounding error, then an iter-ative method like Newton's method or the secant method will converge
until it gets into the interval (2), after which it will behave erati-cally.

The same considerations apply to the function \( g \); however, we
must now consider not only how accurately \( f \) is evaluated but also how
accurately the denominator in (1) is evaluated. Even if the first
function evaluation is done without error, we have for this denominator

\[
\frac{1}{c_p} \frac{1}{c}\left[ \frac{\varepsilon}{c_p}, \frac{\varepsilon}{c} \right]
\]

\[ f[x + f(x)] - f(x) \]

\[ = c[x + cx^p] - cx^p + e[x + f(x)] \]

\[ = c^2 x^{2p-1} \left[ p + \frac{p(p-1)}{2} x^{p-1} + \ldots \right] + e[x + f(x)] \]
As $x$ approaches zero, the true value of the denominator becomes effectively $c^2x^{2p-1}$, and this must be larger than $\epsilon$ to insure that it is at all accurate. In other words one should not expect further convergence once the interval

$$(3) \quad \left[ -\left[ \frac{\epsilon}{c} \right]^{2p-1}, \left[ \frac{\epsilon}{c} \right]^{2p-1} \right]$$

has been entered. For fixed $c$ and sufficiently small $\epsilon$, the interval (3) is always larger than (2).

A numerical experiment, performed on a Texas Instruments Programmable 59 calculator, will illustrate the above considerations. The function $f$ was taken to be

$$f(x) = (x-1)^3 = x^3 - 3x^2 + 3x - 1,$$

which has a triple zero at one. Since the TI 59 carries thirteen decimal digits in its computations, if Horner's method is used to evaluate $f$ near one, we may expect an error on the order of $10^{-13}$. Taking $\epsilon = 10^{-13}$ and $c = 1$, we get for the interval (2) (appropriately translated)

$$(4) \quad 1 + [-0.000046, +0.000046],$$

and for (3)

$$(5) \quad 1 + [-0.0025, +0.0025].$$

The first interval is significantly smaller than the second.
Table 1 shows the results of applying the secant method, starting from $x_1 = 1.4$ and $x_2 = 1.2$, first to $f$ and then to $g$. The convergence for $f$ is slow but steady; the thirtieth iterate is just outside the interval (4), and at that point $f$ evaluates exactly to zero. On the other hand the fourth iterate with the function $g$ is in the interval (5). The iteration could not be carried further because the denominator in (1) evaluated exactly to zero.

The conclusion is that for multiple zeros the $g$-iteration can fail to achieve the maximum accuracy that is attainable at a fixed precision of computation. On the other hand, until the $g$-iteration enters its region of ineffectiveness, its behavior is quite good. In the example, it required the $f$-iteration eighteen function evaluations to produce a solution as accurate as the $g$-iteration produced in six. Perhaps a reasonable compromise is to use the $g$-iteration until it breaks down and then, if further accuracy is required, shift to more refined techniques.

References.

Table 1

f- and g-Iterations for
\[ f(x) = x^3 - 3x^2 + 3x - 1 \]

<table>
<thead>
<tr>
<th>Iterate</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.4000000</td>
<td>1.4000000</td>
</tr>
<tr>
<td>2</td>
<td>1.2000000</td>
<td>1.2000000</td>
</tr>
<tr>
<td>3</td>
<td>1.1714286</td>
<td>0.9438925</td>
</tr>
<tr>
<td>4</td>
<td>1.1228346</td>
<td>1.0016196</td>
</tr>
<tr>
<td>5</td>
<td>1.0945533</td>
<td>---</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>18</td>
<td>1.0024305</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1.0018347</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1.0013850</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>27</td>
<td>1.0001933</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1.0001485</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1.0001150</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1.0000478</td>
<td></td>
</tr>
</tbody>
</table>