Polynomial deflation strategy for root finders.

By Henrik Vestermark (hve@hvks.com)

Abstract:
Several root-finding methods find one or two roots at a time that in turn is deflated into the polynomial and the process is repeated to find one or two more roots until all roots have been found. Now technically you can use either forward deflating or backward deflating or a hybrid composite deflation that combines the advantages of both the forward and backward deflating techniques to preserve the accuracy of the deflated polynomial.

Introduction:

When an iteration step has been completed e.g. a Newton iteration and a root has been found, the root is typically deflated up into the polynomial to reduce the polynomial, and the Newton iterating is repeated until all roots have been found. This paper describes the two methods and in what context it is appropriate to use them in. Not unusually we are seeking the method that preserves the accuracy of the remaining polynomial coefficients to ensure the highest obtainable accuracy of the Newton or similar roots-finding process.

If you have a polynomial with either real or complex coefficients

$$P(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0$$

And a root R (either real or a complex number).
We are trying to find the deflated polynomial that satisfied the equation:

$$\begin{aligned} &P(z) = Q(z)(z - R) \\ &where\ P(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0 \\ &and\ Q(z) = b_{n-1} z^{n-1} + b_{n-2} z^{n-2} + \ldots + b_1 z + b_0 \end{aligned}$$

Now to obtain the b's you can either start by finding the highest coefficient b_{n-1} and work your way down to b_0 which is called *forward* deflation or the opposite find the coefficients starting with b_0 and work your way up to b_{n-1} which is called *backward* deflation.

Forward Deflation of Polynomials:

To do forward deflation we try to solve the equations starting with the highest coefficients a_n:

$$a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0 = (b_{n-1} z^{n-1} + b_{n-2} z^{n-2} + \ldots + b_1 z + b_0)(z - R)$$

The recurrence is given by:

$$
\begin{aligned}
a_n &= b_{n-1} \\
a_k &= b_{k-1} - R * b_k \quad k = n-1,\dots,1 \\
a_0 &= -R * b_0
\end{aligned}
$$

Now solve it for b's you get:

$$
\begin{aligned}
b_{n-1} &= a_n \\
b_k &= a_{k+1} + R * b_{k+1} \quad k = n-2,\dots,0
\end{aligned}
$$

This simple algorithm works well for polynomials with real coefficients and real roots and complex coefficients with complex roots using the same recurrence just using complex arithmetic instead. A special case is real coefficients with complex roots. A complex root and its complex conjugated root will be the same as dividing the polynomial P(Z) with 2nd order polynomial of the two complex conjugated roots (x+*i*y) and (x-*i*y) or (z^2-2xz+(x^2+y^2)). Letting r=-2x and u= x^2+y^2

$$
\begin{aligned}
&P(z) = Q(z)(z^2 + rz + u) \\
&where\ P(z) = a_n z^n + a_{n-1} z^{n-1} + \dots + a_1 z + a_0 \\
&and\ Q(z) = b_{n-2} z^{n-2} + b_{n-3} z^{n-3} + \dots + b_1 z + b_0 \\
&The\ recurrence\ is\ giving\ by: \\
&a_n = b_{n-2} \\
&a_{n-1} = b_{n-3} + rb_{n-2} \\
&a_{n-2} = b_{n-4} + rb_{n-3} + ub_{n-2} \\
&a_2 = b_0 + rb_1 + ub_2 \\
&a_1 = rb_0 + ub_1 \\
&a_0 = ub_0
\end{aligned}
$$

Now solve it for b's you get:

$$
\begin{aligned}
b_{n-2} &= a_n \\
b_{n-3} &= a_{n-1} - r * b_{n-2} \\
b_k &= a_{k+2} - rb_{k+1} - ub_{k+2} \quad k = n-4,\dots,0
\end{aligned}
$$

Backward Deflation of Polynomials:

To do backward deflation we try to solve the equations starting with the lowest coefficient a_0 and work our way up to a_n:

$$a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0 = (b_{n-1} z^{n-1} + b_{n-2} z^{n-2} + \ldots + b_1 z + b_0)(z - R)$$

The recurrence is given by:

$$a_0 = -R * b_0$$
$$a_k = b_{k-1} - R * b_k \quad k = 1, \ldots n-1$$
$$a_n = b_{n-1}$$

Now solve it for b's you get:

$$b_0 = -\frac{a_0}{R}$$
$$b_k = (b_{k-1} - a_k)/R \quad k = 1, \ldots, n-2$$
$$b_{n-1} = a_n$$

For complex conjugated roots, we again divide the quadratic factor (z^2-2xz+(x^2+y^2)) up in the polynomial P(z) this time starting from the back. Letting r=-2x and u= x^2+y^2

$$P(z) = Q(z)(z^2 + rz + u)$$
$$\text{where } P(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0$$
$$\text{and } Q(z) = b_{n-2} z^{n-2} + b_{n-3} z^{n-3} + \ldots + b_1 z + b_0$$
$$\text{The recurrence is giving by}:$$
$$a_0 = ub_0$$
$$a_1 = ub_1 + rb_0$$
$$a_k = ub_k + rb_{k-1} + b_{k-2} \quad k = 2, \ldots, n-2$$
$$a_{n-1} = rb_{n-2} + b_{n-3}$$
$$a_n = b_{n-2}$$

Now solve it for b's and you get

$$b_0 = a_0 / u$$
$$b_1 = (a_1 - r * b_0)/u$$
$$b_k = (a_k - b_{k-2} - rb_{k-1})/u \quad k = 2, \ldots, n-2$$

Forward or Backward Deflation?

Wilkinson [2] has shown that to ensure a stable deflation process you should choose *forward* deflation if you find the roots of the polynomial in increasing magnitude and always deflate the polynomial with the lowest magnitude root first and of course, the opposite *backward* deflation when finding the roots with decreasing magnitude. Although most root-finding algorithms do find them in increasing order it can't be guaranteed and therefore to ensure the most stable deflation process you will use the composite deflation method which is more complicated to handle than the forward or backward deflation technique.

Composite Deflation of Polynomials:

To carry out composite deflation you calculated the new coefficients by doing forward deflations and saving the new coefficients in an array B[]. They do backward deflations and say the new coefficients in an array C[]. You then join the arrays B[] and C[] by finding the coefficients index with the lowest difference in the magnitude between the newly calculated coefficients *k.* You then take the forward deflation coefficients from the B[] from n..k+1 and the backward coefficients C[] from k-1..0 and then take the average for the coefficients k as $b_k = \frac{1}{2}(B[k]+C[k])$.

We then have the algorithm as follows to calculate the new coefficients b's:

```
r=+Infinity
For( i=0..n-1)
        u=|B[i]|+|C[i]|
        If(u!=0) u=|B[i]-C[i]|/u
        If(u<r) u=r, k=i
For(i=k+1..n-1) bi=B[i];
bk= ½ (B[k]+C[k])
For(i=k-1..0) bi=C[i];
```

Grant [1] used the above mention algorithm for composite deflation in solving polynomial equations.

Reference

1. Grant, J A & Hitchins, G D. Two algorithms for the solution of polynomial equations to limiting machine precision. The Computer Journal Volume 18 Number 3, pages 258-264
2. Wilkinson, J H, Rounding errors in Algebraic Processes, Prentice-Hall Inc, Englewood Cliffs, NJ 1963