Stopping criteria for Polynomial root finders
By Henrik Vestermark (hve@hvks.com)

Abstract:
Finding adequate stopping criteria for polynomial root finders is not always easy. To aggressive stopping criteria and you will never converge to an acceptable root or too lax and you find roots with a lesser degree of accuracy than possible by the actual limitation of the machine's precision. Stopping criteria are based on the round-off errors when evaluating a polynomial at a given real or complex point x.

Change log
February 12, 2023. Updated with more content throughout the document. Added and expanded the stopping criteria from Grant-Hitchins, Igarashi, Garwick & Ward, and JL Nikolajsen.

Contents

Introduction:

When locating the zeros of a polynomial, it is usually difficult to know just when to terminate the iteration process. It is desirable to terminate the process when the zero is known to be within the round-off accuracy. Various ad hoc stopping criteria have been used; however, such criteria do not take into account particular properties of the polynomial being evaluated. Such properties might include the condition of the polynomial, multiple zeros, or clusters of zeros. In this paper, a stopping criterion is presented which requires that the value of the polynomial be smaller than a calculated bound for the round-off error.

Before we can jump into a discussion of stopping criteria we first need to see how we evaluate a polynomial at some point z (z can be a real or complex number) and then continue our discussion by first looking at a simple upper bound, following by more advance upper bound for the rounding errors in evaluating a polynomial at some complex or real point.

Evaluation of Polynomials:

To evaluate a polynomial P at z:

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

We generally use Horner recurrence given by the recurrence:

$$b_n = a_n$$
$$b_k = b_{k-1} z + a_k \quad k = n-1,\ldots,0$$

The last term of this recurrence b_0 is now the value of $P(z)$.

Therefore, this evaluation of P(z) requires n multiplications and additions for a total of 2n operations. The above mention recurrence works well for polynomials with real coefficients evaluated at a real point x, as well as for polynomials with complex coefficients evaluated at a complex point Z=x+*i*y in which case multiplication and addition are replaced with the complex multiplication and addition for complex arithmetic given by:

Complex multiplication: (a+*i*b)(c+*i*d) = (ac – bd) + *i*(ad+bc)
Complex addition: (a+ib)+(c+id) = ac + *i*bd

Since a Complex multiplication requires 4 ‘real’ multiplications and 2 additions the total number of operations involving is 4n+2n or 6n ‘real’ operations for polynomials with complex coefficients evaluated at a complex point.

In the case of a polynomial P with real coefficients evaluated at a complex point Z we in general are using Horner recurrence but in a special version using only real arithmetic:

$$Z = x + iy$$
$$p = -2x$$
$$q = x^2 + y^2$$

$$b_n = a_n$$
$$b_{n-1} = a_{n-1} - pb_n$$
$$b_k = a_k - pb_{k+1} - qb_{k+2} \qquad k = n-2,\ldots,1$$
$$b_0 = a_0 + xb_1 - qb_2$$
$$P(Z) = b_0 + iyb_1$$

It, therefore, requires a 4n operation instead of 2n for the real case to evaluate a polynomial with real coefficients and a complex point Z.

Polynomial	Real coefficient	Complex coefficients
The Number of operations:		
Real point	2n	**4n**
Complex point	**4n**	**6n**

Algorithm Horner with real coefficients at a real point

```
// Evaluate a polynomial with real coefficients a[] at a real point r
// and return the result in fz
// the function returns the absolute magnitude of the evaluation
double horner(const int n, const double a[], const double r, double *fz)
	{
	double fval;

	fval = a[0];
	for (int i = 1; i <= n; i++)
		fval = fval * r + a[i];

	*fz = fval;
	return abs(fval);
	}
```

Algorithm Horner with real coefficients at a complex point

```
// Evaluate a polynomial with real coefficients a[] at a complex point z
// and return the result in fz
// the function returns the absolute magnitude of the evaluation
double horner( const int n, const double a[], const complex<double> z,
complex<double> *fz )
   {
   int i;
```

```
   double p, q, r, s, t;

   p = -2.0 * z.real();
   q = norm( z );
   s = 0; r = a[ 0 ];
   for( i = 1; i < n; i++ )
      {
      t = a[ i ] - p * r - q * s;
      s = r;
      r = t;
      }
   *fz = complex<double>( a[ n ] + z.real() * r - q * s, z.imag() * r );

   return abs( *fz );
   }
```

Algorithm Horner with complex coefficients at a complex point

```
// Evaluate a polynomial with complex coefficients a[] at a complex point z
// and return the result in fz
// the function returns the absolute magnitude of the evaluation
double horner( const int n, const complex<double> a[], const complex<double> z,
complex<double> *fz )
   {
   complex<double> fval;

   fval = a[ 0 ];
   for( int i = 1; i <= n; i++ )
      fval = fval * z + a[ i ];

   *fz = fval;
   return abs( fval );
   }
```

Error in arithmetic operations:

J.H.Wilkinson in “Rounding errors in algebraic processes” [3] has shown that the errors in performing algebraic operations are bound by:

$\varepsilon < \frac{1}{2}\beta^{1-t}$ β *is the base, and t is the precision* (Assuming round to nearest)

For the Intel microprocessor series and the IEE754 standard for floating point operations $\beta = 2$ and t=53 for 64bit floating point arithmetic or 2^{-53}

A simple upper bound:

A simple upper bound can then be obtained using the above information for a polynomial with degree n.

Polynomial		
The number of operations:	Real coefficient	Complex coefficients
Real point	$\|a_o\|\cdot 2n\cdot 2^{-53}$	$\|a_o\|\cdot 4n\cdot 2^{-53}$
Complex point	$\|a_o\|\cdot 4n\cdot 2^{-53}$	$\|a_o\|\cdot 6n\cdot 2^{-53}$

A better upper bound.

In this category, we have among others Adams [1] and Grant & Hitchins [2] stopping criteria for polynomials.

Polynomial root finders usually can handle polynomials with both real and complex coefficients evaluated at a real or complex number. In principle, we have 3 different scenarios (real coefficients at a real point, real coefficients at a complex point, and complex coefficients at a complex point) that we must deal with to calculate a root to the limitations of the machine precision. Since the bound of the round-off errors are different for these 3 scenarios we need to evaluate them individually.

Case 1: Stopping criteria

Polynomial with real coefficients a_n evaluated at a real point x, using Horner's method:

$$b_n = a_n$$
$$b_k = b_{k-1}x + a_k \quad k = n-1,\ldots,0$$

An error bound can be computed using a similar recurrence as follows, see Kahan[7]:

$$e_n = |b_n|\frac{1}{2}$$
$$e_k = e_{k-1}|x| + |b_k| \quad k = n-1,\ldots,0$$
$$e = (4e_0 - 2|b_0|)\varepsilon \quad where\ \varepsilon = \frac{1}{2}\beta^{1-t}$$

Algorithm Kahan Stopping Criteria

```
// Calculate the upper bound for the rounding errors performed in a
// polynomial with real coefficient a[] at a real point z. Kahan
//
double upperbound(const int n, const double a[], const double r)
       {
       int i;
       double t, e;
```

```
      t = a[0]; e = abs(t)*(0.5);
      for (i = 1; i<n; i++)
            {
            t = t*r + a[i];
            e = abs(r)*e + abs(t);
            }
      e = (2 * e - abs(t))*pow((double)_DBL_RADIX, -DBL_MANT_DIG + 1);

      return e;
      }
```

Case 2: Adams Stopping Criteria

Using Horner's method, a polynomial with real coefficients is evaluated at a complex point z.

$$Z = x + iy$$
$$p = -2x$$
$$q = x^2 + y^2$$
$$b_n = a_n$$
$$b_{n-1} = a_{n-1} - pb_n$$
$$b_k = a_k - pb_{k+1} - qb_{k+2} \qquad k = n-2,...,1$$
$$b_0 = a_0 + xb_1 - qb_2$$
$$P(Z) = b_0 + iyb_1$$

Adams [1] has shown that an error bound can be computed using the following recurrence:

$$e_n = |b_n|\frac{7}{9}$$
$$e_k = e_{k-1}|Z| + |b_k| \quad k = n-1,...,0$$
$$e = (4.5e_0 - 3.5(|b_0| + |b_1||Z|) + |x||b_1|)\varepsilon \quad where\ \varepsilon = \frac{1}{2}\beta^{1-t}$$

Algorithm Adams Stopping Criteria

```
// Calculate an upper bound for the rounding errors performed in a
// polynomial with real coefficient a[] at a complex point z. ( Adam's test )
//
double upperbound( const int n, const double a[], const complex<double> z )
   {
   int i;
   double p, q, r, s, t, u, e;

   p = - 2.0 * z.real();
   q = norm( z );
   u = sqrt( q );
```

```
   s = 0.0; r = a[ 0 ]; e = fabs( r ) * ( 3.5 / 4.5 );
   for( i = 1; i < n; i++ )
      {
      t = a[ i ] - p * r - q * s;
      s = r;
      r = t;
      e = u * e + fabs( t );
      }
   t = a[ n ] + z.real() * r - q * s;
   e = u * e + fabs( t );
   e = ( 4.5 * e - 3.5 * ( fabs( t ) + fabs( r ) * u ) +
       fabs( z.real() ) * fabs( r )) * 0.5 * pow( (double)_DBL_RADIX, -
DBL_MANT_DIG+1);

   return e;
   }
```

Case 3: Grant & Hitchins stopping criteria

Using Horner's method, a polynomial with complex coefficients zn is evaluated at a complex point z. This gets a little bit more complicated. Grant and Hitchins [2] derive an upper error bound for the errors in evaluating the polynomial as follows

$$P(Z) = (a_n + ib_n)z^n + (a_{n-1} + ib_{n-1})z^{n-1} + \ldots + (a_1 + ib_1)z + (a_0 + b_0)$$

Using Horner's method and keeping track of the real component c_k and the imaginary component d_k of the coefficient separately we get:

$$\begin{aligned} c_n &= a_n, \quad d_n = b_n \\ c_k &= c_{k+1}x - yd_{k+1} + a_k \quad k = n-1,\ldots,0 \\ d_k &= d_{k+1}x + yc_{k+1} + b_k \quad k = n-1,\ldots,0 \end{aligned}$$

Using these values an error bound can now be calculated using the recurrence:

$$\begin{aligned} g_n &= 1, \quad h_n = 1 \\ g_k &= |x|(g_{k+1} + |c_{k+1}|) + |y|(h_{k+1} + |d_{k+1}|) + |a_k| + 2|c_k| \quad k = n-1,\ldots,0 \\ h_k &= |y|(g_{k+1} + |c_{k+1}|) + |x|(h_{k+1} + |d_{k+1}|) + |b_k| + 2|d_k| \end{aligned}$$

Now the error is $(g_0+ih_0)\varepsilon$, where $\varepsilon = \frac{1}{2}\beta^{1-t}$. Now since the recurrence in itself introduce error [2] safeguard the calculation by adding the upper bound for the rounding errors in the recurrence, so we have the bound for evaluating a complex polynomial in a complex point:

$$e = (g_0 + ih_0)\varepsilon(1+\varepsilon)^{5n} \quad where\ \varepsilon = \frac{1}{2}\beta^{1-t}$$

Other methods in this category are Igarshi's, Garwick's, and Ward's. The nice parts of these stopping criteria are that they don't discriminate whether the polynomial is with real or complex coefficients at a real or complex point as long as the calculation is done with proper respect for the type of coefficient and the type of evaluation point.

Algorithm Grant & Hitchins Stopping Criteria

```
// Calculate an upper bound for the rounding errors performed in a
// polynomial with complex coefficient a[] at a complex point z. ( Grant &
Hitchins test )
//
double upperbound(const int n, const complex<double> a[], complex<double> z )
	{
	int i;
	double nc, oc, nd, od, ng, og, nh, oh, t, u, v, w, e;
	double tol = 0.5* pow((double)_DBL_RADIX, -DBL_MANT_DIG + 1);

	oc = a[0].real();
	od = a[0].imag();
	og = oh = 1.0;
	t = fabs(z.real()); u = fabs(z.imag());
	for (i = 1; i <= n; i++)
		{
		nc = z.real() * oc - z.imag() * od + a[i].real();
		nd = z.imag() * oc + z.real() * od + a[i].imag();
		v = og + fabs(oc); w = oh + fabs(od);
		ng = t * v + u * w + fabs(a[i].real()) + 2.0 * fabs(nc);
		nh = u * v + t * w + fabs(a[i].imag()) + 2.0 * fabs(nd);
		og = ng; oh = nh;
		oc = nc; od = nd;
		}
	e = abs(complex<double>(ng,nh) ) * pow(1 + tol, 5 * n) * tol;
	return e;
	}
```

Igarashi's

Igarashi's suggested back in 1984 a new stopping criterion for finding the roots of the polynomial P(z).

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

Igarashi's suggested a stopping criterion after the i'th iteration when:

$$|P(z_i) - B(z_i)| \geq \min(|P(z_i)|, |B(z_i)|)$$

Where $B(z) = zP'(z) - C(z)$ and $C(z) = zP'(z) - P(z)$. Of course, they have to be evaluated before the subtraction and you get the following two evaluations calculated using the Horner methods.

$$zP'(z) = na_n z^n + (n-1)a_{n-1}z^{n-1} + \cdots + a_1 z$$
$$C(z) = (n-1)a_n z^n + (n-2)a_{n-1}z^{n-1} + \cdots + a_2 z^2 - a_0$$

Initially, when you are far from the root the $|P(z_i) - B(z_i)|$ will be smaller than $\min(|P(z_i)|, |B(z_i)|)$, however as you approach the root both $P(z_i)$ and $B(z_i)$ will go towards zero but then $|P(z_i) - B(z_i)|$ will be dominated by the round-off errors and become larger than $\min(|P(z_i)|, |B(z_i)|)$ providing a suitable stopping criteria for the root search.

Igarashi suggests that the search will terminate if one of the 3 conditions arises:

a) If $P(z_i)$ *or* $B(z_i) = 0.0$
b) If $P(z_i)B(z_i) < 0$
c) If $P(z_i)B(z_i) > 0$ and ($2|P(z_i)| \leq |B(z_i)|$ or $2|B(z_i)| \leq |P(z_i)|$)

Algorithm Igarashi with real coefficients at a real point

```
// Igarashi stopping criteria for Polynomial with real coefficients
// at a real point r
// n is the degree of the polynomial
// Notice that a[0] is an, a[1] is an-1 and a[n]=a0
//
bool Igarashi(const int n, const double a[], const double r)
	{
	double *zP = new double[n+1];
	double *C = new double[n+1];
	double px, zpx, cx, bx;

	for (int i = 0; i <= n; i++)
		{
		zP[i] = (n - i) * a[i];
		C[i] = (n - i - 1) * a[i];
		}
	horner(n, a, r, &px);
	horner(n, zP, r, &zpx);
	horner(n, C, r, &cx);
	bx = zpx - cx;
	delete [] zP, C;
	if (px == 0.0 || bx == 0.0) return true;
	if (px*bx < 0) return true;
	if (2 * fabs(px) <= fabs(bx) || 2 * fabs(bx) <= fabs(px)) return true;
	return false;
	}
```

Algorithm Igarashi with real coefficients at a complex point

```
// Igarashi stopping criteria for Polynomial with real coefficients
// at a complex point z
// n is the degree of the polynomial
// Notice that a[0] is an, a[1] is an-1 and a[n]=a0
//
```

```
bool Igarashi(const int n, const double a[], const complex<double> z)
	{
	double *zP = new double [n + 1];
	double *C = new double [n + 1];
	complex<double> px, zpx, cx, bx;

	for (int i = 0; i <= n; i++)
		{
		zP[i] = (double)(n - i) * a[i];
		C[i] = (double)(n - i - 1) * a[i];
		}
	horner(n, a, z, &px);
	horner(n, zP, z, &zpx);
	horner(n, C, z, &cx);
	bx = zpx - cx;
	delete[] zP, C;
	if (px == 0.0 || bx == 0.0) return true;
	if (px.real()*bx.real() < 0 || px.imag() * bx.imag() < 0) return true;
	if (2 * abs(px) <= abs(bx) || 2 * abs(bx) <= abs(px)) return true;   return
false;
	}
```

Algorithm Igarashi with complex coefficients at a complex point

```
// Igarashi stopping criteria for Polynomial with complex coefficients
// at a complex point z
// n is the degree of the polynomial
// Notice that a[0] is an, a[1] is an-1 and a[n]=a0
//
bool Igarashi(const int n, const complex<double> a[], const complex<double> z )
	{
	complex<double> *zP = new complex<double> [n + 1];
	complex<double> *C = new complex<double> [n + 1];
	complex<double> px, zpx, cx, bx;

	for (int i = 0; i <= n; i++)
		{
		zP[i] = (double)(n - i) * a[i];
		C[i] = (double)(n - i - 1) * a[i];
		}
	horner(n, a, z, &px);
	horner(n, zP, z, &zpx);
	horner(n, C, z, &cx);
	bx = zpx - cx;
	delete[] zP, C;
	if (px == 0.0 || bx == 0.0) return true;
	if (px.real()*bx.real() < 0 || px.imag() * bx.imag() < 0 ) return true;
	if (2 * abs(px) <= abs(bx) || 2 * abs(bx) <= abs(px)) return true;
	return false;
	}
```

Garwick's & Ward's

Garwick (see JLN[5]) introduces this very simple stopping criterion that states that when the increment from two iterative steps $e_i > e_{i-1}$, where $e_i = |z_i - z_{i-1}|$ then the root z_{i-1} is found. When convergence has first started then the rate of convergence does not

decrease until a root has been found. Ward (see JLN[5]) improve on the initial problem with the Garwick precondition issue and states the following stopping criterion:

$$z_{i-1}\ is\ a\ root\ if\ e_i > e_{i-1}, where\ e_i = |z_i - z_{i-1}|$$

JLN [5] replace it to:

$$z_{i-1}\ is\ a\ root\ if\ e_i \geq e_{i-1}, where\ e_i = |z_i - z_{i-1}|$$

after numerical results show Ward originally failed to stop under certain conditions.

And the following preconditions hold:

$$(1) \quad e_i \leq 10^{-7}\ if\ |z_{i-1}| < 10^{-4}$$

$$(2) \quad \frac{e_i}{|z_{i-1}|} \leq 10^{-3} if |z_{i-1}| \geq 10^{-4}$$

Algorithm Garwick & Ward with real coefficients at a real point

```
// Garwick stopping criteria.
// r, r1 & r2 is the 3 latest root estimations.
// Convergence rate only decrease due to rounding errors then
// we continue until the new r has a larger step size than the previous
// r1 (due to round-off errors)
// Return true if stopping criteria have been reached otherwise false
//
bool Garwick(const double r, const double r1, const double r2 )
      {
      double e1, e2;

      e1 = fabs(r - r1);  // Newest stepsize
      e2 = fabs(r1 - r2); // Previous stepsize
      if( fabs(r1) < 1E-4 && e1 <= 1E-7 ||
          fabs( r1 ) >= 1E-4 && e1/fabs(r1)<=1E-3)
             if (e1 >= e2) return true;
      return false;
      }
```

Algorithm Garwick & Ward with complex coefficients at a complex point

```
// Garwick stopping criteria.
// z, z1 & z2 is the 3 latest root estimations.
// Convergence rate only decrease due to rounding errors then
// we continue until the new z has a larger step size than the previous
// z1 (due to round-off errors)
// Return true if stopping criteria have been reached otherwise false
//
bool Garwick(const complex<double> z, const complex<double> z1, const
complex<double> z2)
      {
      double e1, e2;

      e1 = abs(z - z1);  // Newest stepsize
      e2 = abs(z1 - z2); // Previous stepsize
```

```
        if (abs(z1) < 1E-4 && e1 <= 1E-7 ||
            abs(z1) >= 1E-4 && e1 / abs(z1) <= 1E-3 )
               if (e1 >= e2) return true;
        return false;
        }
```

An even better upper bound

JL Nikolajsen [5] write an excellent paper and suggested a new stopping criterion for iterative root finding. His suggestion eliminates unnecessary function evaluations and also immediately stop the iterations when no further improvement to the roots is possible. JLN outlines 4 possible stopping criteria capable of also handling the ill-conditioned root. The method works equally well for both real and complex roots. Instead of repeating the JLN finding I will just summarize the 4 different stopping criteria

JLN Sopping criterion 1

$$z_i \; is \; a \; root \; if \; \frac{s_i^2}{s_{i-1}} \geq s_m$$
$$Precondition: s_{i-1} \geq \frac{s_m}{q_m^2}$$

S_i is the number of matching leading bits (MLBs) of the two successive iterates z_{i-1} and z_i, s_m is the length of the IEEE 754 floating point double precision e.g. S_m=53 bits, and q_m is the convergence order of the iterative method used. E.g. Newton is 2, Halley is 3 and Laguerre is also 3, etc.

JLN Stopping criterion 2

$$z_{i+1} \; is \; a \; root \; if \; \frac{s_i^2}{s_{i-1}} > s_{i+1}$$
$$Preconditions: s_{i-1} \geq \frac{s_m}{q_m^2} \; and \; s_i - s_{i-1} \geq \frac{s_m}{q_m^2}$$

This stopping criterion is used when the criterion 1 convergence rate is not quite fast enough to trigger the stopping criterion 1.

JLN Stopping criterion 3

Stopping criterion 3 is used to catch stop after a single iteration if needed and comes in 2 sub-criteria

$$z_i \; is \; a \; root \; if$$
$$1\text{: } z_0 \neq 0 \; and \; s_1 \geq \frac{s_m}{2}$$
$$2\text{: } z_0 = 0 \; and \; s_1 \geq s_m$$

$$z_i \; is \; a \; root \; if$$
$$1\text{: } s_i - s_{i-1} \geq \frac{s_m}{2} \;\; or$$
$$2\text{: } s_1 - s_{i-1} \geq \frac{s_m}{4} \; and \; s_{i+1} - s_i < s_i - s_{i-1} \; when \; i \geq 2$$

JNL Stopping criterion 4

The last stopping criterion is.

$$z_{i+1} \; is \; a \; root \; if s_{i+2} \leq s_{i+1} \; with \; the \; precondition\text{:}$$
$$s_{i-1} \geq b, s_i \geq b \; and \; b = 8$$

As already mention I encourage readers to study the JLN method [5] in detail and JLN more elaborated explanation and details of the method.

Other methods:

[4] Provide a comprehensive list of other methods to consider and is a good reference for what has been done in this field over the last many years.

Interval arithmetic is another obvious choice. The benefit is that if we use interval arithmetic in our evaluation we immediately have a bound for the error in our evaluation. The stopping criteria will be if the polynomial evaluation using interval arithmetic contains an interval containing zero.

Reference

1. Adams, D A stopping criterion for polynomial root finding. Communication of the ACM Volume 10/Number 10/ October 1967 Page 655-658
2. 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
3. Wilkinson, J H, Rounding errors in Algebraic Processes, Prentice-Hall Inc, Englewood Cliffs, NJ 1963
4. McNamee, J.M., Numerical Methods for Roots of Polynomials, Part I, Elsevier, Kidlington, Oxford 2009
5. Jorgen L. Nikolajsen New Stopping criteria for iterative root finding, Royal Society Open Science, 16 September 2014 http://dx.doi.org/10.1098/rsos.140206
6. Igarashi 1989, A termination criterion for iterative methods used to find the zeros of polynomials. Mathematical Computation, Volume 42, Page 165-171.
7. Kahan W and Farkas I, Algorithm 168 and Algorithm 169. Comm. ACM 6 (Apr. 19G3), 165.