Improving polynomial evaluation at an approximate root

D. Westreich

Department of Computer Engineering, MBT, Israel Aircraft Industries Ltd, Yahud, Israel

A reduced round off error procedure is given for determining the accuracy of an approximate root of a polynomial.

(Received July 1979)

When considering high degree polynomials with large coefficients it is difficult to determine if an approximate root returned by a polynomial root solver subroutine is sufficiently near the actual root. One would usually accept the root if the value of the polynomial at the root is near zero. However, when evaluating the polynomial at a large root on a computer, the nature of the polynomial together with floating point arithmetic are apt to introduce large round off errors so that no significant results may be obtained. To overcome this difficulty we recommend the following simple expedient.

Suppose $P(z) = \sum_{i=0}^{n} a_i z^i$ is an n degree polynomial, $a_0 \neq 0$ and z_0 is an approximate root of $P(z)$. Let $[x]$ be the greatest integer function and define

$j = 1$ if n is odd and $|z_0| > 1$

$j = 0$ otherwise.

Then the roots of $P(z)$ are equivalent to those of

$$F(z) = P(z)/z^k = Q(z) + R(z)$$

where $k = [n/2] + j$

$$Q(z) = \sum_{i=k}^{n} a_i z^{i-k}$$

and

$$R(z) = \sum_{i=0}^{k-1} a_i z^{i-k}$$

$Q(z)$ is a polynomial of degree $n - k$ ($\approx n/2$) and $R(z)$ is a polynomial in $1/z$ of degree k ($\approx n/2$). Thus, instead of evaluating $P(Z_0)$ we evaluate $Q(z_0)$ and $R(z_0)$ where in each case the round off error will be considerably less than for $P(z_0)$. We will then accept the root z_0 if $Q(z_0) + R(z_0)$ is sufficiently near zero.

This principle may be used to improve the accuracy of an approximate root. That is, use the Newton-Raphson iteration scheme applied to $F(z)$, i.e. $z_{i+1} = z_i - F(z_i)/F'(z_i)$ starting at z_0.

Permutation groups and set union algorithms

M. D. Atkinson

Department of Computing Mathematics, University College, Cardiff

In this note we give one more application of the set union techniques described in Aho, Hopcroft and Ullman (1974) for representing a partition of a set, testing for membership of a subset and replacing two parts by their union. Other applications are given there but ours is to a completely different area: computational group theory.

Suppose that we are given a permutation group H on the symbols $1, 2, \ldots, n$, i.e. a set of one to one functions (permutations) defined on $1, 2, \ldots, n$ which is closed under functional composition. In this situation the symbols may be partitioned into orbits, two symbols being in the same orbit if one is mapped on to the other by one of the permutations of H.

Suppose also that we are given one more permutation g and that G is the group obtained by composing g with the permutations of H in all possible ways (G is the group generated by H and g). Then the orbits of G can be obtained from those of H by successively combining subsets of $1, 2, \ldots, n$ according to the following scheme:

initially $1, 2, \ldots, n$ is partitioned into subsets given by the orbits of H;
for each $\alpha = 1, 2, \ldots, n$
  let β be the image of α under g;
  if the subset containing α is different from the subset containing β then replace these subsets by their union.

It is easily proved that the resulting partition of $1, 2, \ldots, n$ is the set of orbits of G.

Clearly the above process can be implemented by one of the standard set union algorithms. The best one currently known has a running time which grows only slightly faster than n and so the orbits of G can be computed in an acceptably short time.

Finally, notice that if a group is given by generating permutations $g_1, g_2, \ldots$ we can obtain its orbits starting from the trivial partition of $1, 2, \ldots, n$ into singleton subsets and successively using the above with $g = g_1, g_2, \ldots$ This method of computing orbits does not require all generators to be simultaneously present in main memory (unlike the one described by McKay and Regner (1974) and so may offer advantages when storage space is limited.

References

Aho, A. V., Hopcroft, J. E. and Ullman, J. D. (1974). *The Design and Analysis of Computer Algorithms*, Addison-Wesley.
McKay, John and Regner, E. (1974). Algorithm 482, Transitivity Sets, *CACM*, Vol. 17 No. 8 p. 470.