

Re: High Precision Approximate GCD

Source: <http://sci.tech-archive.net/Archive/sci.math.symbolic/2004-07/0197.html>

From: Dave Rusin (rusin_at_vesuvius.math.niu.edu)

Date: 07/23/04

Date: 23 Jul 2004 18:56:23 GMT

In article <EGaMc.16317\$8_6.5658@attbi_s04>, Z. Zeng <zzeng@neu.edu> wrote:

```

>Let
>
>p = (x-1)^i*(x-2)^j*(x-3)^k*(x-4)^l
>q = p'
>
>GCD(p,q) = (x-1)^(i-1) * (x-2)^(j-1) * (x-3)^(k-1) * (x-4)^(l-1)
>
>For different sets of [i,j,k,l], the following table lists the error in
>computed approximate GCD's
>using hardware precision (16 digits)
>
> [i,j,k,l] EpsilonGCD QRGCD UvGCD*
>-----
> 2,1,1,0 Fail 1e-13 7e-16

```

Let me see if I understand this: you have taken the polynomial p, presumably in expanded form $x^4-7x^3+17x^2-17x+6$, and its derivative $4x^3-21x^2+34x-17$, and one of these algorithms `_failed_` to find a gcd from these data?

As I wrote privately to the OP, one can view the search for the gcd as a set of linear equations to be solved, which is certainly a pretty well-understood area of numerical analysis. In the OP's case, there was a clear understanding that the gcd was to be of degree 1 but the technique is similar whenever the degree of the gcd is known (obviously not always an easy thing to discover or even to define). We simply want to find polynomials P and Q of suitable degrees so that $pP + qQ$ is a monic polynomial of the specified degree. In this example, a linear gcd requires $\deg(Q)=3$ and $\deg(P)=2$. So here it is in Maple:

```

P:=add( a[i] * x^(2-i), i=0..2):
Q:=add( b[i] * x^(3-i), i=0..3):
total := expand( p*P + q*Q - (x^1 + junk) ):
eqs:={ seq( coeff( total, x, i ), i= 1 .. 6 ) }:
answer:=solve(eqs):

```

```
taylor( subs(answer,p*P+q*Q) , x, 7 );
```

The result is $-1 + x$, which is the correct answer. It is simple to adapt this code to other degrees for p and q if the degree of the gcd is also known.

I don't deny that the "solve" command masks quite a bit of complexity which would be present when the the coefficients are floating-point values or when the polynomials are only `_approximately_` equal to multiples of the gcd, but it is rather astounding to me that there would be software which would fail with such simple initial data. Could you elaborate? Moreover, could you explain why the naive approach I have suggested is not a good basis for solving problems like the OP had?

FWIW, I tried more subtle cases like the ones the OP proposed. I randomly picked one root, picked five more with the kinds of variation s/he proposed, and held to the condition that the linear coefficient be zero.

Starting with roots $r1=0.2491072884$ and $s1 = 0.2491079210$, respectively, for the two polynomials, and using random leading coefficients as well,

I was led to these two quartics:

$p:=0.4689144526*x^4 -0.3119802476*x^3 +.05837848017*x^2 -.0006056607140:$

$q:=.09329956241*x^4 -.06206752682*x^3 +.01161291099*x^2 -.0001204532483:$

whose roots are, respectively,

$-0.08316541550, 0.2491371046, 0.2494694059, 0.2498833494$

$-0.08315593602, 0.2491118087, 0.2491901349, 0.2501039165$

that is, the round-off error while computing the coefficients of the expanded forms of p and q had moved the roots around nearly as much as the distance to the other roots the OP was `_not_` interested in.

At this point I believe it is very much a judgement call to decide what the degree of the gcd is ! Insisting, as the OP did, that the gcd

be of degree 1, the code I showed gives a gcd of approximately

$x - 0.2491426331$

when I stick to Maple's default precision for the calculations and blithely ignore the extra degree of freedom in the "solve" step.

(There are ways to make a concrete choice but I don't know which is "best".)

dave