

# Simplifying RootSum Expressions

Richard Fateman  
Computer Science  
University of California, Berkeley

June 10, 2009

## Abstract

It's useful to sum an expression with a parameter varying over all the roots of a given polynomial. Here's a defense of that statement and a method to do the task.

## 1 Rational Integration and RootSum

Maxima as well as other computer algebra systems can sometimes return symbolic integration results as expressions including a construction which is the sum of  $f(a)$  where  $a$  runs through all roots of an algebraic equation  $p(a) = 0$ . We refer to this as a `RootSum` term<sup>1</sup>, although the name is not entirely standard.

This construction comes about when a rational function is integrated, but the answer cannot be expressed in terms of explicit radicals. In Maxima we can obtain the following kind of answer by suitably setting a flag:

```
integrate_use_rootsof:true$  
integrate(1/(x^3+a*x+1),x)$
```

returns

$$\sum_{r \in \text{rootsof}(r^3+ar+1, r)} \frac{\log(x-r)}{a+3r^2}$$

differentiation wrt  $x$  should give:

$$\sum_{r \in \text{rootsof}(r^3+ar+1, r)} \frac{1}{(a+3r^2)(x-r)}$$

Now to see if this is the right answer, we should be able to simplify this expression, but Maxima does not have a built-in simplifier for this case. Neither does the Macsyma commercial version.

The program `SRS` (for simplify RootSum) below does so.

## 2 How to write the program

Writing programs in Maxima's user language rather than in the underlying Lisp language makes it easier for more people to read (or modify) the code.

While we ordinarily have promoted writing in Lisp where language semantics are clearer (not dependent on complicated operations like "simplify"), in this case the operations needed are pretty-much what Maxima provides, and the cost of interpreting the short guidance code is expected to be negligible relative to the much higher cost of doing the embedded polynomial arithmetic implied by calls to the library.

---

<sup>1</sup>Actually, a construction `lsum` is used which in the current 5.18 version interferes with the processing. In subsequent versions, currently in progress, `lsum` should work fine. We altered the programs for differentiation, integration, and added a second parameter to `rootsof`

### 3 First we must rationalize denominators

Our SRS program requires that we have a polynomial function of  $a$ , and so we must first take the summand and rationalize the denominator: it must not include  $a$ .

For this we apply the Extended Euclidean Algorithm (EEA), where one available in Maxima is called `gcdex`.

Given  $u$  and  $v$  EEA solves for  $A$  and  $B$  in  $Au + Bv = 1$ , but modulo  $v$ ,  $Bv$  is 0 and so  $A$  is  $1/u$ . In the process,  $A$  comes out with a rationalized denominator. Here's the program:

```
/*RD will return n/d with a rationalized denominator free of r, given
   that poly(r)=0 */
```

```
RD(n,d,poly,r):=
  part(gcdex(d,poly,r),1)*n$
```

### 4 Main program

Now we are ready for the main program, which requires us to compute a sum of a polynomial or rational function<sup>2</sup> in  $r$  (and perhaps other parameters) where  $r$  runs over the set of roots  $\{r_1, r_2, \dots\}$ . To do this we need values for  $r_1 + r_2 + \dots$ ,  $r_1^2 + r_2^2 + \dots$ ,  $r_1^3 + r_2^3 + \dots$ , etc.

Fortunately, these symmetric functions of the roots can be expressed entirely within the representation of the coefficients of the roots of expression. The technique is given by the so-called “Newton identities” or the Newton-Girard formulas<sup>3</sup>, and is expressed through the array of `e` values in the SRS program below.

```
/* Simplification of RootSum expressions... SRS(q,p,r) */
```

```
/* q is a polynomial or rational function in (among other things),
   the 3rd parameter, r.
   p is a polynomial in r,
   defining a set of roots r1,r2, .. such that p(ri)=0
   r is the variable used in the expressions q and p.
```

```
SRS, simplify rootsum, does this: It computes the sum of q(r) over the roots.
That is, it computes q(r1)+q(r2)+ ...+q(rn).
```

```
Example: SRS (3*r+r^2, (r-a)*(r-b)*(r-c)*(r-d),r) is equivalent to
3*(a+b+c+d) + (a^2+b^2+c^2+d^2). */
```

```
SRS(q,p,r):=
  block([qn,d,lc],
    local(e,w),
    p:rat(p,r), /* make p into CRE form for speed in hipow and ratcoef */
    q:RD(ratnumer(q:rat(q,r)),ratdenom(q),p,r), /*rationalize denominator of q*/
    qn: ratnumer(q), /* numerator of q, after rationalization */
    d:hipow(p,r), /* degree of p*/
    lc:ratcoef(p,r,d), /* leading coefficient */
    e[i]:=ratcoef(p,r,d-i)/lc,
    /* compute newton identities */
    w[0]:d, /*w[deg] is sum of roots^deg */
```

<sup>2</sup>By rationalizing the denominator, we reduced rational functions to polynomials: we take  $r$  out of the denominator and then take the denominator out of the sum.

<sup>3</sup>The details of which I found in Wikipedia

```
w[1]:rat(-e[1]),
w[deg]:=-sum(e[j]*w[deg-j],j,1,deg-1)-deg*e[deg],
/* put pieces together and divide by denom [which is free of r] */
sum(w[i]*ratcoef(qn,r,i),i,0,hipow(qn,r))/ratdenom(q))$
```

There are a few spare parts that need to be added to the Maxima system, which we have done in several minor fixes for versions after 5.18.

Completing the example given earlier, `SRS(1/((a+3*r^2)*(x-r)),(r^3+a*r+1),r)` indeed gives back the expected result  $1/(x^3+ax+1)$ .

In general, we can define a function to simplify the result of integration and then differentiation, if that results in some left-over `lsum` pieces, and the summand is a rational function in the algebraic parameter (which it probably is!) Let the expression be `INTDIF`. Applying this function should return the original integrand.

```
simp_lsum(INTDIF):=ev(INTDIF, nounify(lsum)= lambda([a,b,c],SRS(a,part(c,1),b)));
```

## 5 Thanks

Thanks to Barry Trager for reminding me of what I had forgotten, or should have known but did not.

## 6 Appendix: programs only

```
RD(n,d,poly,r):= part(gcdex(d,poly,r),1)*n$

/* see paper rootsum.pdf or rootsum.tex for documentation */
SRS(q,p,r):=
  block([qn,d,lc],
    local(e,w),
    p:rat(p,r), /* make p into CRE form for speed in hipow and ratcoef */
    q:RD(ratnumer(q:rat(q,r)),ratdenom(q),p,r), /*rationalize denominator of q*/
    qn: ratnumer(q), /* numerator of q, after rationalization */
    d:hipow(p,r), /* degree of p*/
    lc:ratcoef(p,r,d), /* leading coefficient */
    e[i]:=ratcoef(p,r,d-i)/lc,
    /* compute newton identities */
    w[0]:d, /*w[deg] is sum of roots^deg */
    w[1]:rat(-e[1]),
    w[deg]:=-sum(e[j]*w[deg-j],j,1,deg-1)-deg*e[deg],
    /* put pieces together and divide by denom [which is free of r] */
    sum(w[i]*ratcoef(qn,r,i),i,0,hipow(qn,r))/ratdenom(q))$

matchdeclare([m,r,u],true)$
tellsimpafter('diff('rootsum(m,r,u),r,1),rootsum(diff(m,r),r,u))$
/* atest, should be 1/(x^3+a*x+1)
atest: SRS(1/((a+3*r^2)*(x-r)),(r^3+a*r+1),r)
btest: SRS(r,(a*r-b)*(r-c)*(r-d),r)
should be b/a+c+d
```

If you are dealing with the output of `integrate INTDIF`, differentiated, try this ...

```
simp_lsum(INTDIF):=subst(lambda([a,b,c],SRS(a,part(c,1),b)),nounify(lsum),INTDIF);  
*/
```