Polynomial Algebra in Form 4

J Kuipers[1]

[1] Nikhef, Science Park 105, 1098 XG Amsterdam, The Netherlands

E-mail: jkuipers@nikhef.nl

Abstract. New features of the symbolic algebra package Form 4 are discussed. Most importantly, these features include polynomial factorization and polynomial gcd computation. Examples of their use are shown. One of them is an exact version of Mincer which gives answers in terms of rational polynomials and 5 master integrals.

1. Introduction

Form is a symbolic algebra package that has been developed by J. Vermaseren et al. for the past decades [1]. Its fast manipulation of large expressions makes it an ideal tool to attack the big problems that occur in particle phenomenology. Recently, version 4β has been released, that includes many new features including polynomial algebra. This new feature can be divided in three different operations, namely greatest common divisors (gcds), factorization and rational polynomials. This paper discusses these operations and provides examples of Form code to show how to use them.

2. Greatest common divisors

The greatest common divisor of two polynomials is a polynomial that divides both of them and that is divisable by every other polynomial that divides both. In Form 4, this polynomial can be determined by function `gcd_` in the following way.

```
Symbols x,y;
Local E = gcd_(x^2+x*y, y^2+x*y);
Print;
.end

E=
   y + x;
```

In previous version of Form, the `gcd_` function would return `E=1` for this particular example.

To determine the gcd, Form uses the following algorithms. For small polynomials, a heuristic that substitutes integers and performs integer gcd calculations is used [2]. For large polynomials, Zippel's modular algorithm is used [3]. This latter algorithm is fast for both sparse and dense polynomials. The speed of Form's gcd algorithms is comparable to Mathematica's.

3. Factorization

In Form, there are a few statements that can be used to factorize a polynomial over the rational numbers. First, there exists `FactArg_`, which is used to factorize a function argument. Its usage is as follows.

```
Symbols k,i,f,e,h;
CFunction N;
Local E = N(k*f+k*e+k*h+i*f+i*e+i*h);
FactArg N;
Print;
.end

E=
  N(i + k,h + e + f);
```

Note that this changes the previous behaviour of `FactArg_`. To get the old behaviour, a switch `On OldFactArg` has been implemented.

The second variety of polynomial factorization is for dollar variables and can be invoked as follows.

```
Symbols x,y;
#$a = x^2-y^2;
#Factorize $a;
#do i=1, `$a[0]'
  #write "%$", $a[`i'];
#enddo

-y+x
y+x
```

Note that `$a[0]` results in the number of factors, while `$a[i]` yield the actual factors.

The third way of using the factorization routines works for expressions. The usage is as follows:

```
Symbols x;
Local E = x^2-1;
Factorize E;
.end

E =
    ( -1 + x )
    * ( 1 + x );
```

The number of factors of a factorized expression is given by `numfactors(E)` and the individual factors can be addressed to by `E[factor_^i]`.

To factorize polynomials, Form uses the following algorithms. For univariate polynomials, Berlekamp's algorithm is used [4]. Multivariate polynomials are reduced to univariate ones by substuting integers for all but one variable and factorized as such. Afterwards, the multivariate result is obtained by Hensel lifting [5]. For big random polynomials that factorize into two factors, speedups of upto a factor of 1 000 have been measured compared to Mathematica.

4. Rational polynomials

In Form, the statement `PolyRatFun` turns a function into a rational polynomial with the first argument as numerator and the second argument as denominator. The function also serves as coefficient of the terms, just like the `PolyFun` in previous versions of Form. The following code shows how it can be used.

```
Symbols x,y,z;
CFunction f;
PolyRatFun f;
Local E = x * f(y,z) + x * f(y,1-z)
        + x^2 * f(y^2-1,y-1);
Print;
.end

E =
    x*f( - y,z^2 - z) + x^2*f(y + 1,1);
```

5. Application: MincerExact

As a first application of Form's polynomial algebra, a new version of Mincer [6] has been written. Mincer is a program for three-loop massless propagator diagrams and it works in expansions in $\epsilon = (4 - D)/2$. The original version keeps track of expansions typically up to the sixth power. Big tables with expansions of Pochhammer symbols, inverse gamma functions and alike are needed for the calculations.

The new version, MincerExact, does not expand in ϵ, but stores the result as a rational polynomial in it. In this way, no big tables are needed, which results in much shorter and cleaner source code. The resulting program is not much slower than the original version.

References

[1] Vermaseren J A M 2000 New features of FORM *Preprint* math-ph/0010025v2
[2] Char B W, Geddes K O and Gonnet G H 1989 *J. Symbolic Comp.* **9** 31–48
[3] Zippel R E 1979 *Probabilistic algorithms for sparse polynomials* (M.I.T. PhD thesis)
[4] Berlekamp E R 1967 *Bell System Technical Journal* **46** 1853–1959
[5] Geddes K O, Czapor S R, and Labahn G 1992 *Algorithms for Computer Algebra* (Boston: Kluwer)
[6] Larin S A, Tkachov F V and Vermaseren J A M 1991 The FORM version of Mincer *Preprint* NIKHEF-H/91-18