Notes on `libfpoly`

Frank James*

May 30, 2013

1 Algorithms and data structures

Polynomials are implemented as flat arrays of integer coefficients. The coefficients may be fixed precision integers (fixnums) or high precision integers (bignums).

Coefficients in the array can be accessed using the function `offset` defined below.

```
(defun base-offset (num-vars degree)
  (if (< degree 0)
      0
      (ncr (+ num-vars degree)
           degree)))

(defun power-offset (powers)
  (let ((degree (reduce #'+ (cdr powers))))
    (cond
      ((or (null (cdr powers))
           (= (+ (car powers) degree) 0))
       0)
      (t
        (+ (number-terms (length powers) (1- degree))
        (power-offset (cdr powers)))))))

(defun offset (powers)
  (+ (base-offset (length powers) (1- (reduce #'+ powers)))
     (power-offset powers)))
```

Note that the total number of coefficients N is given by

$$N = \frac{(n+m)!}{n!}, \tag{1}$$

with n the degree of the polynomial and m the number of variables.

*email: frank.a.james@gmail.com

1.1 Systems of linear equations

Systems of linear equations with polynomial coefficients can be solved using the fraction free Gaussian elimination (FFGE) algorthim. This can be implemented for either integers or polynomials in general.

There are two approaches to implementing FFGE.

1. Directly implementing FFGE for matrices of polynomial (or integer) entries

2. Choosing a set of evaluation points for all the variables in the polynomial entries and then evaluating the matrix at each of these points. This results in a set of matrices of numbers only, which can then be solved using FFGE in the standard way. These can then be combined back into the polynomial solutions using Lagrange interpolation

In both of these cases, bignum integers are required. This can be overcome by first choosing a set of prime numbers and taking the input matrix modulo these. Then either of the above processes can be used to generate solution matrices for each of the primes. Finally, these can be combined back into the complete solution using the Chinese Remainder theorem. Using this procedure, only fixnum integers are required.

2 Issues

2.1 Problems with polynomial FFGE

The FFGE algorithm causes the degree of the polynomials to increase. The polynomial arithmetic functions addition/subtraction scale as the total number of coefficients N whereas multiplication scales as N^2. The situation with division is less clear since it depends on the density of the dividend/divisor.

Let us see how N scales as n, m,

$$\frac{N_{n+1,m}}{N_{n,m}} = 1 + \frac{m}{n+1}, \frac{N_{n,m+1}}{N_{n,m}} = \frac{n+m+1}{.} \tag{2}$$

Clearly increasing the degree of a polynomial only slowly increases the total number of coefficients. However, increasing the number of variables very quickly increases the number of coefficients. For polynomials of reasonably large degree (20) this is clearly a very large increase.

2.2 Probiems with evaluation/interpolation

The FFGE algorithm scales as r^3 with r the number of unknowns to solve for in the input linear system. This is the rank of the input matrix.

Interpolation on the otherhand requires computing determinants of NxN matrices. Determinants can be computed using a fraction free LU decomposition algorithm, which is essentially a generalisation of FFGE, and also scales as

n^3. However, in this case, the rank of the input matrix is not r but N. As shown above, N can become very large rather quickly and the work required to compute these determinants becomes far more than is required to reduce the input matrix.

3 Conclusion

Both approaches have their merits and unfortunately both suffer from drawbacks. In the case of the evaluation/interpolation approach, the drawback seems rather serious.