TITLE:

Simultaneous field divisions: an extension of Montgomery's trick

AUTHOR: David G. Harris

Department of Defense

9206 Daleview Court

Silver Spring, MD USA 20901

davidgharris29@hotmail.com

ABSTRACT:

Montgomery's trick is a technique which can be used to quickly compute multiple field inversions simultaneously. We extend this technique to simultaneous field divisions (that is, combinations of field multiplications and field inversion). This generalized Montgomery's trick is faster in some fields than a simple inversion with Montgomery's trick followed by a simple field multiplication.

KEYWORDS: Montgomery's trick, simultaneous inversion, simultaneous division

1 Introduction

Field inversions are typically expensive to compute, requiring far more operations than simpler computations such as addition and multiplication. For example, inversion in a prime field requires computing an Extended GCD. If we have many independent inversions to compute, i.e. we want to compute

$$\frac{1}{y_i} \qquad i = 1, \ldots, N$$

then the algorithm known as Montgomery's trick can be used to produce all N inverses simultaneously, at the cost of just one field inversion and $3N - 2$ field multiplications. In Section 2, we will review this algorithm.

Often, however, the field inversion is part of a larger arithmetic computation. Suppose we wish to compute

$$\frac{c}{y_i} \qquad \text{or} \qquad \frac{c\ x_i}{y_i}$$

Here, c is some constant multiplier applied to all of the N computations, and x_i (optional) is some numerator that varies among the N separate computations. For example, to double an elliptic curve point (X, Y) in affine coordinates, one needs to compute

$$\lambda = \frac{3X^2 + a}{2Y}$$

where a is a constant parameter of the elliptic curve.

In this case, the simplest approach would be to use the numerator $x_i = 3X^2 + a$, the denominator $y_i = Y$, and the constant multiplier $c = 1/2$. As we will see, a better approach is to set $x_i = X^2 + (a/3)$, $y_i = Y$, and the constant multiplier $c = 3/2$.

Of course, we could use Montgomery's trick to compute all the denominators $1/y_i$, and then multiply by the numerators c and x_i, at the cost of additional field multiplications. There have been many applications of Montgomery's trick to elliptic curve arithmetic, such as [1] and [2], and they apparently have all followed this plan.

However, there is an alternative, which is to modify Montgomery's trick to incorporate the multiplication of x_i and c with the field inversion. In Section 4, we will describe how to do this. Because our algorithm incorporates a numerator as well as a denominator, we refer to this as a field *division* algorithm, as opposed to merely field *inversion*.

2 Montgomery's trick for multiple field inversions

We will first review Montgomery's trick for simultaneous field inversions as described in [1]. We are given N field elements $y_i \in F$, and wish to compute $1/y_i$ for $i = 1, \ldots, N$. We will do this by performing two separate "passes" through the data. In the forward pass, we initialize $r_1 = y_i$ and compute the forward products

$$r_i = r_{i-1} \times y_i, \qquad i = 2, \ldots, N$$

When this is done, we have $r_N = \prod_{i=1}^{N} y_i$. We compute a single field inversion

$$I = r_N^{-1}$$

Next, we enter the backward pass. To begin, we set $t_N = I$ and then for

$i = N, \ldots, 2$, we compute the two products

$$1/y_i = t_i \times r_{i-1}$$

$$t_{i-1} = t_i \times y_i$$

To finish, we set

$$1/y_1 = t_1$$

In total, this algorithm requires a single field inverse; in the forward pass, it requires $N - 1$ field multiplications; and in the backward pass, it requires $2N - 2$ field multiplications.

One observation has been apparently overlooked or at least underestimated: the backward pass's two multiplications share a common factor. That is, instead of performing two unrelated multiplications

$$a \times b \qquad c \times d$$

we are performing two related multiplications

$$a \times b \qquad a \times c$$

For some fields, it may be faster to compute these two related multiplications as compared to unrelated multiplications. We will discuss this more in Section 3. For the moment, let us simply summarize the cost of this algorithm as

$$\text{Cost} = (N - 1) \times M_1 + (2N - 2) \times M_2 + \text{Inversion}$$

where M_1 is the cost of an ordinary field multiplication; M_2 is half the cost of a pair of "double multiplications", i.e. multiplications involving a shared multiplicand; and Inversion is the cost of a field inversion.

To simplify the costing, Cost/N as $N \to \infty$ goes to

$$\text{Cost}/N \to M_1 + 2M_2$$

3 Double multiplication

Depending on the specific field, computing a double multiplication

$$a \times b \qquad a \times c$$

may be less expensive than computing two unrelated multiplications. In costing field arithmetic, it is well-known that squaring may be cheaper than a general multiplication. By the same token, these double multiplications may be cheaper too and should be accounted separately. The reason is that many methods of multiplication are based on "transforming" one or both multiplicands. In a double multiplication, we need only transform a once.

For example, in very large prime fields, (e.g. 2000+ bits), field multiplication is generally structured as first an ordinary integer multiplication followed by a modular reduction. For the integer multiplication one views the multiplicands as integer polynomials, which one Fourier transforms, multiplies pointwise, and then inverse transforms the product. A double multiplication saves a transform compared to two ordinary multiplications.

As another example, in small prime fields $\mathbf{F}_p$ (e.g. 100—400 bits) one typically represents the multiplicands in "Montgomery form." In this case, if the radix size of the computer is W, we represent a by $a_0, \ldots, a_{k-1}$ such that

$$a = W^k(a_0 + a_1 W + a_2 W^2 + \cdots + a_{k-1} W^{k-1}) \mod p$$

To double-multiply a by b and c, we can use the Montgomery multiplication formula

$$ab = \sum_{i=0}^{k-1}((aW^{-i}) \bmod p) \times b_{k-1-i}$$
$$ac = \sum_{i=0}^{k-1}((aW^{-i}) \bmod p) \times c_{k-1-i}$$

Again, we are "transforming" a by successively dividing it by powers of W; and this transformation can be shared between the two products.

In some fields, a double multiplication is not noticeably faster than two ordinary multiplications. For example, in medium prime fields $\mathbf{F}_p$, multiplication is most efficiently computed by Karatsuba multiplication followed by modular reduction or Montgomery reduction. The Karatsuba algorithm does not take advantage of the common factor.

In general, then, we can say that double multiplication can sometimes be faster, for some fields, than ordinary multiplication.

4 Simultaneous field divisions: constant numerator

To return to the problem at hand, suppose we wish to compute something more complicated than a simple field inversion. At first, let us analyze the computation

$$c/y_i \qquad i = 1, \ldots, N$$

Using the inversion algorithm, we could compute the denominators $1/y_i$ at a per-unit cost $M_1 + 2M_2$. We could then multiply by the constant numerator

c to get a total cost of

$$\text{Cost} = N \times M_1 + (2N - 2) \times M_2 + \text{Inversion} + NM_c$$

As $N \to \infty$, the per-unit cost is

$$\text{Cost}/N \to M_1 + 2M_2 + M_c$$

Note that we are treating multiplication by c differently than an ordinary multiplication. In many applications, c is small or has special structure making multiplication by c significantly cheaper than a general field multiplication. For example, in the case of elliptic curve affine point doubling, $c = 3/2$. We can multiply by $3/2$ with just a few bit-shifts and field additions, so the cost to multiply by c will be small, although not totally negligible.

We will now show how to combine the multiplications by c with the multiple inversion algorithm, resulting in a lower cost. As before, we have a forward pass, a field inversion, and a backward pass.

To begin, we initialize the forward pass with $r_1 = y_1$, and then for $i = 2, \ldots, N$ compute

$$r_i = r_{i-1} \times y_i$$

When this is done, we have $r_N = \prod y_i$. We next compute

$$I = c \times r_N^{-1}$$

Next, we enter the backward pass. To begin, we set $t_N = I$ and then for

$i = N, \ldots, 2$ we compute the two products

$$c/y_i = t_i \times r_{i-1}$$

$$t_{i-1} = t_i \times y_i$$

and finish by

$$c/y_1 = t_1$$

In total, the cost of the foward pass is $(N-1) \times M_1$; we then perform a modular inversion and a single multiplication by c; finally, the backward pass costs $(N-1) \times 2M_2$. In total, the cost is

$$\text{Cost} = (N-1)M_1 + (2N-2)M_2 + M_c + \text{Inversion}$$

As $N \to \infty$, the per-unit cost tends to

$$\text{Cost}/N \to M_1 + 2M_2$$

Note that the multiplication by c has been completely amortized away, just like the field inversion. Although the cost of multiplying by c may be small, it is not negligible. A side benefit is that, even if c has a special form, there is no need for specialized computer code to multiply by it since it is no longer a time-critical step.

5 Simultaneous field divisions: variable numerator

Now let us consider the case of a variable numerator as well as variable denominator. We now wish to compute

$$c\ x_i/y_i \qquad i = 1, \ldots, N$$

Using the algorithm of Section 3, we could compute the fractions c/y_i at a per-unit cost $M_1 + 2M_2$. We could then multiply by the numerators x_i to get a total cost of

$$\text{Cost} = 2NM_1 + (2N-2)M_2 + \text{Inversion}$$

As $N \to \infty$, the per-unit cost is

$$\text{Cost}/N \to 2M_1 + 2M_2$$

We will now show how to interleave the multiplications by x_i and c with the multiple inversion algorithm, resulting in a lower cost. As before, we have a forward pass, a field inversion, and a backward pass. All these stages are changed however.

To begin, we start the forward pass by setting

$$r_1 = y_1$$

and then for $i = 2, \ldots, N$ we compute

$$r_i = r_{i-1} \times y_i$$
$$s_i = r_{i-1} \times x_i$$

Note that the two multiplications share a common factor, and thus this is a double multiplication.

When this is done, we have $r_N = \prod y_i$. We next compute

$$I = c \times r_N^{-1}$$

Next, we enter the backward pass. To begin, we set $t_N = I$ and then for $i = N, \ldots, 2$ we compute the two products

$$c\ x_i/y_i = t_i \times s_i$$
$$t_{i-1} = t_i \times y_i$$

and finally finish with

$$c\ x_1/y_1 = t_1 \times x_1$$

In total, the cost of the foward pass is $(N-1) \times 2M_2$; we then perform a modular inversion and a single multiplication by c; finally, the backward pass costs $M_1 + (N-1) \times 2M_2$. In total, the cost is

$$\text{Cost} = M_1 + (4N-4)M_2 + M_c + \text{Inversion}$$

As $N \to \infty$, the per-unit cost tends to

$$\text{Cost}/N \to 4M_2$$

As compared to our previous cost, we have replaced the two unrelated multiplications by a double multiplication.

6 Conclusion

Inversion in a field can be quite costly. For this reason, if many field inversions need to be computed, Montgomery's trick is a useful technique for replacing them all with a single field inversion and many multiplications.

In many cases, such as elliptic curve affine point doubling, these field inversions go along with field multiplications, a combination we define as a

field division. We have described an algorithm which incorporates this numerator into the Montgomery's trick, resulting in a computation which is faster than the simple Montgomery's trick in two cases: 1) If the numerator involves a constant term; or 2) If double multiplication (a pair of multiplications with common factor) can be computed more quickly than two unrelated multiplications.

7 Acknowledgements

Thanks to Joana Silva for helping edit this paper, and for helping me with the publication process.

8 Reference

[1] H. Cohen, "A Course in Computational Algebraic Number Theory." Graduate Texts in Math. 138, Springer-Verlage, 1993.

[2] P. Mishra, S. Palash, "Application of Montgomery's trick to scalar multiplication for elliptic and hyperelliptic curves using a fixed base point." PKC 2004, March 2004, pp. 41-54.