Long Integers and Polynomial Evaluation with Estrin's Scheme

Marco Bodrato

mambaSoft
Via S.Marino, 118
10137 Torino, Italy

bodrato@mail.dm.unipi.it

Alberto Zanoni

Dipartimento di Scienze Statistiche
Università "Sapienza"
P.le Aldo Moro 5 - 00185 Roma, Italy

zanoni@volterra.uniroma2.it

Universitatea de Vest din Timişoara

SYNASC 2011

September 26th, 2011 - Timişoara, Romania

Summary

1. **Polynomial evaluation**
   - Notation and costs
   - Multiplication algorithms and complexity
   - Ruffini-Horner method
2. **Estrin approach**
   - Description
   - Comparing Ruffini-Horner and Estrin
   - Threshold issues
3. **Variants, unbalancedness, sparsity**
   - Estrin variants: F and BZ
   - Thresholds and sparse polynomials
   - Conclusions

Long integer and polynomial evaluation

Some notation and costs

Problem: Evaluate a polynomial in a long integer x with $|x| \gg 1$

- $p(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{Z}[x] \quad : \quad d = \deg(p) \quad ; \quad D = d + 1.$
  Size of a_i and x: $\simeq [\log_2(a_i)] \simeq [\log_2(x)] = n$

Long integer and polynomial evaluation

Some notation and costs

Problem: Evaluate a polynomial in a long integer x with $|x| \gg 1$

- $p(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{Z}[x] \quad : \quad d = \deg(p) \quad ; \quad D = d + 1.$
  Size of a_i and x: $\simeq [\log_2(a_i)] \simeq [\log_2(x)] = n$
- Consider two long integers with m, n digits in base-2 representation (bits), respectively:

  Operation costs:

  $M(m,n)$: multiplication $\qquad M(n) = M(n,n)$
  $A(m,n)$: addition/subtraction $\qquad A(n) = A(n,n)$

  One can assume $A(m,n) = A(\min(m,n))$

Polynomial evaluation

Costs

- Ruffini-Horner : d multiplications, d additions

$$p(x) = ((\cdots((a_d x + a_{d-1})x + a_{d-2})x + \cdots)x + a_1)x + a_0$$

- Motkin'55, Belaga'61, Pan'66 : by preconditioning, around $d/2$ multiplications are sufficient
- Paterson, Stockmeyer '73 : $\mathrm{O}(\sqrt{d})$ multiplications

Polynomial evaluation

Costs

- Ruffini-Horner : d multiplications, d additions

$$p(x) = ((\cdots((a_d x + a_{d-1})x + a_{d-2})x + \cdots)x + a_1)x + a_0$$

- Motkin'55, Belaga'61, Pan'66 : by preconditioning, around $d/2$ multiplications are sufficient
- Paterson, Stockmeyer '73 : $\mathrm{O}(\sqrt{d})$ multiplications

Above complexities are measured just counting the *number* of multiplications (i.e. considering every product having constant cost). For "growing" factors (e.g. long integers), this is not sufficient to understand global complexity.

Evaluation of polynomials

Costs

Papers considering x and/or a_i as long integers:

- Akritas, Danielopulos '80 : polynomial translation
- Danielopulos '82 : polynomial and derivatives evaluation

...but only "schoolbook" $\mathrm{O}(n^2)$ multiplication is considered.

Actually there are many different subquadratic multiplication methods. What happens if they are used ?

Multiplication algorithms

Many algorithms are known for long integer multiplication.

- Schoolbook $O(n^2)$

Each one has a different complexity, and its own range where it is the fastest one.

Multiplication algorithms

Many algorithms are known for long integer multiplication.

- Schoolbook $O(n^2)$
- Karatsuba (1962) $O(n^{\log_2 3})$

Each one has a different complexity, and its own range where it is the fastest one.

Multiplication algorithms

Many algorithms are known for long integer multiplication.

- Schoolbook — $O(n^2)$
- Karatsuba (Toom-2) (1962) — $O(n^{\log_2 3})$
- Toom-Cook-k (1963) — $O(n^{\log_k (2k-1)})$

Each one has a different complexity, and its own range where it is the fastest one.

Multiplication algorithms

Many algorithms are known for long integer multiplication.

- Schoolbook $O(n^2)$
- Karatsuba (Toom-2) (1962) $O(n^{\log_2 3})$
- Toom-Cook-k (1963) $O(n^{\log_k (2k-1)})$
- Schönhage-Strassen (1971) $O(n \log n \log \log n)$

Each one has a different complexity, and its own range where it is the fastest one.

Multiplication algorithms

Many algorithms are known for long integer multiplication.

- Schoolbook — $O(n^2)$
- Karatsuba (Toom-2) (1962) — $O(n^{\log_2 3})$
- Toom-Cook-k (1963) — $O(n^{\log_k (2k-1)})$
- Schönhage-Strassen (1971) — $O(n \log n \log\log n)$
- Fürer (2007) — $O(n \log n 2^{\log^* n})$

Each one has a different complexity, and its own range where it is the fastest one.

Multiplication algorithms

Many algorithms are known for long integer multiplication.

- Schoolbook $O(n^2)$
- Karatsuba (Toom-2) (1962) $O(n^{\log_2 3})$
- Toom-Cook-k (1963) $O(n^{\log_k (2k-1)})$
- Schönhage-Strassen (1971) $O(n \log n \log \log n)$
- Fürer (2007) $O(n \log n 2^{\log^* n})$

Each one has a different complexity, and its own range where it is the fastest one.

Balanced approach : factors have the same number of bits (n)
Unbalanced approach : $m \neq n$ [B., Z. '07 : Toom-$(k+1/2)$]

Detailing Ruffini-Horner method

Unbalanced multiplications appear

Ruffini-Horner $\Rightarrow$

```
i = d; result = a_i;
while(i > 0) do
   i ← i - 1;
   result ← result · x ;
   result ← result + a_i;
```

Detailing Ruffini-Horner method

Unbalanced multiplications appear

Ruffini-Horner $\Rightarrow$

i = d; result = a_i;
while(i $>$ 0) **do**
  i $\leftarrow$ i - 1;
  result $\leftarrow$ result $\cdot$ x;
  result $\leftarrow$ result + a_i;

result	grows by $\sim [\log_2(x)]$ bits at every iteration
x	does not grow

More and more unbalanced multiplication

Possibility of using subquadratic methods is not fully exploited ☹

Estrin's scheme (1960) - augmenting parallelism

Splits $p(x)$ focusing on power of 2 exponents. Let $\Delta = 2^{\lfloor \log_2 d \rfloor}$:

$$p(x) = \left(\sum_{i=\Delta}^{d} a_i x^{i-\Delta} \right) x^\Delta + \left(\sum_{i=0}^{\Delta-1} a_i x^i \right) = p_1(x) x^\Delta + p_0(x)$$

The same approach is applied recursively to $p_0(x)$ and $p_1(x)$.

d	$p(x)$
2	$(a_2)x^2 + a_1 x + a_0$
3	$(a_3 x + a_2)x^2 + a_1 x + a_0$
4	$(a_4)x^4 + (a_3 x + a_2)x^2 + a_1 x + a_0$
5	$(a_5 x + a_4)x^4 + (a_3 x + a_2)x^2 + a_1 x + a_0$
6	$((a_6)x^2 + a_5 x + a_4)x^4 + (a_3 x + a_2)x^2 + a_1 x + a_0$
7	$((a_7 x + a_6)x^2 + a_5 x + a_4)x^4 + (a_3 x + a_2)x^2 + a_1 x + a_0$
8	$(a_8)x^8 + ((a_7 x + a_6)x^2 + a_5 x + a_4)x^4 + (a_3 x + a_2)x^2 + a_1 x + a_0$
$\vdots$	$\vdots$

Estrin's scheme (example)

Two computations: **1)** Products **2)** Successive squares of x

Case $d = 7$ Products (and sums, too) Squares

$$A_3^{(1)} = a_7 x + a_6 \quad A_2^{(1)} = a_5 x + a_4 \quad A_1^{(1)} = a_3 x + a_2 \quad A_0^{(1)} = a_1 x + a_0$$

$$A_1^{(2)} = A_3^{(1)} x^2 + A_2^{(1)} \qquad A_0^{(2)} = A_1^{(1)} x^2 + A_0^{(1)}$$

$$A_0^{(3)} = A_1^{(2)} x^4 + A_0^{(2)}$$

$x \rightarrow x^2 \rightarrow x^4$

Estrin's scheme (example)

Two computations: **1)** Products **2)** Successive squares of x

Case $d = 7$ Products (and sums, too) Squares

$a_7 \quad a_6 \quad a_5 \quad a_4 \quad a_3 \quad a_2 \quad a_1 \quad a_0$

$$A_3^{(1)} = a_7 x + a_6 \qquad A_2^{(1)} = a_5 x + a_4 \qquad A_1^{(1)} = a_3 x + a_2 \qquad A_0^{(1)} = a_1 x + a_0$$

$$A_1^{(2)} = A_3^{(1)} x^2 + A_2^{(1)} \qquad A_0^{(2)} = A_1^{(1)} x^2 + A_0^{(1)}$$

$$A_0^{(3)} = A_1^{(2)} x^4 + A_0^{(2)}$$

Squares: $x \downarrow x^2 \downarrow x^4$

Products are now balanced.
Subquadratic methods can be more profitably applied ☺

Ruffini-Horner (RH) versus Estrin

Compare multiplication complexities for evaluation:

$$E_{RH} = \sum_{i=1}^{D-1} M(in, n) \simeq \sum_{i=1}^{D-1} iM(n, n) = M(n) \sum_{i=1}^{D-1} i = M(n)\frac{D(D-1)}{2}$$

$$E_E = E_E^{(p)} + E_E^{(s)}$$

With Toom-Cook methods

in Estrin's scheme one obtains…

$$\begin{bmatrix} M(kn) \simeq (2k-1)M(n) \\ S(kn) \simeq (2k-1)S(n) \end{bmatrix}$$

Let $\alpha = \log_k(2k-1)$ and $D = 2^\delta$: product complexity

$$E_E^{(p)} \simeq \sum_{i=0}^{\delta-1} \frac{D}{2^{i+1}}(2k-1)M\left(\frac{2^i n}{k}\right) \simeq \frac{D}{2}\sum_{i=0}^{\delta-1}\frac{(2k-1)^2}{2^i}M\left(\frac{2^i n}{k^2}\right) \simeq \cdots$$

$$\simeq \frac{D}{2}\sum_{i=0}^{\delta-1}\frac{(2k-1)^h}{2^i}M\left(\frac{2^i n}{k^h}\right) = \left[k^h = 2^i \Rightarrow h = i\log_k 2\right] =$$

$$= \frac{D}{2}\sum_{i=0}^{\delta-1}\frac{(2k-1)^{i\log_k 2}}{2^i}M(n) =$$

$$= M(n)\frac{D}{2}\sum_{i=0}^{\delta-1}\left(\frac{(2k-1)^{\log_k 2}}{2}\right)^i = [\alpha = \log_k(2k-1)] =$$

$$= M(n)\frac{D}{2}\frac{(2^{\alpha-1})^\delta - 1}{2^{\alpha-1}-1} = M(n)\frac{D}{2}\frac{D^{\alpha-1}-1}{2^{\alpha-1}-1}$$

Let $\alpha = \log_k(2k-1)$ and $D = 2^\delta$: squaring complexity

$$E_E^{(s)} = \sum_{i=0}^{\delta-2} S(2^i n) \simeq \sum_{i=0}^{\delta-2} (2k-1) S\left(\frac{2^i n}{k}\right) \simeq \cdots$$

$$\simeq \sum_{i=0}^{\delta-2} (2k-1)^h S\left(\frac{2^i n}{k^h}\right) = \left[k^h = 2^i \Rightarrow h = i \log_k 2\right]$$

$$= \sum_{i=0}^{\delta-2} \left[(2k-1)^{\log_k 2}\right]^i S(n) = S(n) \sum_{i=0}^{\delta-2} (2^\alpha)^i =$$

$$= S(n) \frac{(2^\alpha)^{\delta-1} - 1}{2^\alpha - 1} = \frac{S(n)}{2^\alpha - 1}\left[\left(\frac{D}{2}\right)^\alpha - 1\right]$$

For the complexity of $S(n)$ we can write $S(n) = \mathrm{O}(M(n))$ the result is

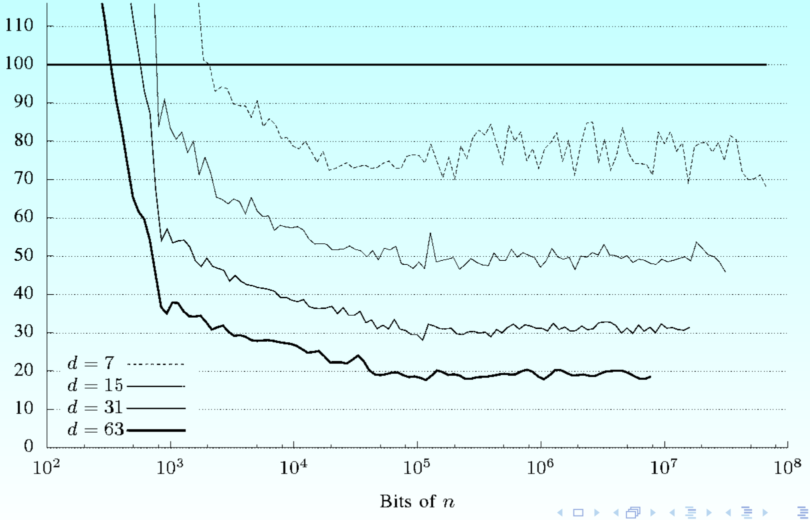
$$E_{RH} = \mathrm{O}(M(n) \cdot D^2) \simeq \mathrm{O}(M_{quadratic}(Dn))$$
$$E_E = \mathrm{O}(M(n) \cdot D^\alpha) \simeq \mathrm{O}(M_{fast}(Dn))$$

Where α is the exponent given by the sub-quadratic multiplication algorithm used.

If coefficients a_i are “small” – $\mathrm{O}(1)$ bits – The product costs slightly changes, but the order of magnitude doesn’t.

Next slide: timings using PARI/GP to compare performances

Graphical comparison: Estrin/RH timings (%)



Estrin convenience threshold $(d = 2)$

Basic case with Ruffini-Horner $\left[(a_2 x + a_1)x + a_0\right]$

$$\begin{aligned} E_{RH} &= (M(n) + A(n)) + M(2n, n) + A(n) \\ &\simeq M(2n, n) + M(n) + 2A(n) \end{aligned}$$

Estrin $\left[a_2(x^2) + (a_1 x + a_0)\right]$ asks instead for

$$\begin{aligned} E_E &= (M(n) + A(n)) + (S(n) + M(2n, n)) + A(2n) \\ &\simeq M(2n, n) + M(n) + 2A(n) + S(n) + A(n) \end{aligned}$$

If we keep on assuming $a_i \simeq x$, RH is better, for odd degrees.

Estrin method: the F (fusion) variant

Example : If $D = 2^\delta + 1$ then x^{2^δ} is computed "only" to multiply a_{2^δ} by it. Can we *skip* the head coefficient?

d	$p(x)$
2	$(a_2)x^2 + a_1x + a_0$
3	$(a_3x + a_2)x^2 + a_1x + a_0$
4	$(a_4)x^4 + (a_3x + a_2)x^2 + a_1x + a_0$
5	$(a_5x + a_4)x^4 + (a_3x + a_2)x^2 + a_1x + a_0$
6	$((a_6)x^2 + a_5x + a_4)x^4 + (a_3x + a_2)x^2 + a_1x + a_0$
7	$((a_7x + a_6)x^2 + a_5x + a_4)x^4 + (a_3x + a_2)x^2 + a_1x + a_0$
8	$(a_8)x^8 + ((a_7x + a_6)x^2 + a_5x + a_4)x^4 + (a_3x + a_2)x^2 + a_1x + a_0$
$\vdots$	$\vdots$

Estrin method: the F (fusion) variant

Example : If $D = 2^\delta + 1$ then x^{2^δ} is computed "only" to multiply a_{2^δ} by it. Can we *skip* the head coefficient? Yes...

$$p'(x) = (a_d x + a_{d-1})x^{d-1} + \cdots + a_0 = a'_{d-1}x^{d-1} + \cdots + a_0$$

Generalization : split $p(x)$ with $\Delta' = \lfloor \log(d+1) \rfloor - 1$

If $d = 2^\delta - 1$ then $\Delta = \Delta'$ (Estrin $\equiv$ F)

d	$p(x)$
2	$(a_2x+a_1)x + a_0$
3	$(a_3x+a_2)x^2 + a_1x + a_0$
4	$((a_4x+a_3)x+a_2)x^2 + a_1x + a_0$
5	$((a_5x+a_4)x^2+a_3x+a_2)x^2 + a_1x + a_0$
6	$(((a_6x+a_5)x+a_4)x^2+a_3x+a_2)x^2 + a_1x + a_0$
7	$((a_7x+a_6)x^2+a_5x+a_4)x^4+(a_3x+a_2)x^2 + a_1x + a_0$
8	$(((a_8x+a_7)x+a_6)x^2+a_5x+a_4)x^4+(a_3x+a_2)x^2 + a_1x + a_0$
$\vdots$	$\vdots$

Estrin method: beyond F variant

Is F always convenient ? As a'_d size can be different, study again the basic case with different coefficient sizes.

Let $Ay^2 + By + C$ be the expression to be evaluated.

$$RH: \; (A \cdot y + B) \cdot y + C$$
$$E: \; A \cdot y^2 + B \cdot y + C$$

Let $\text{size}(y) = n$, $\text{size}(A) = a$, $\text{size}(B) = b$

Consider products and squares only:

$$E_{RH} = M(a, n) + M(\max(a+n, b), n)$$
$$E_E = S(n) + M(a, 2n) + M(b, n)$$

Estrin method: the BZ ("biasing zest") variant

We assume $M(\alpha,\beta) = L(\alpha+\beta)$. Two possibilities for E_{RH}:

1) $a+n \leqslant b$: then

$$E_{RH} \simeq L(a+n)+L(b+n) \leqslant L(a+2n)+L(b+n)+S(n) \simeq E_E$$

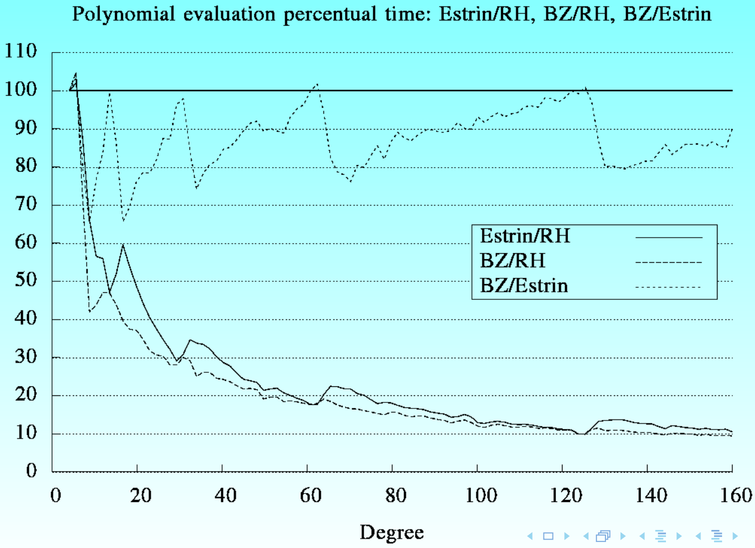
$\Rightarrow$ Ruffini-Horner is faster.

2) $a+n > b$: then

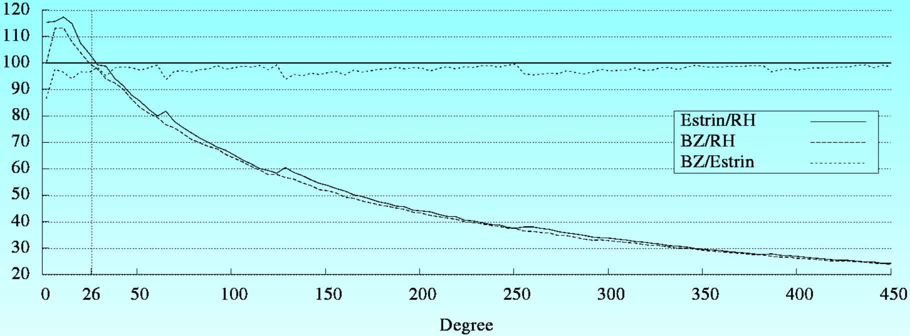
$$E_{RH} \simeq L(a+n)+L(a+2n)$$

$\Rightarrow$ If $S(n)$ has already been computed, one must compare $L(a+n)$ and $L(b+n)$: if $a \leqslant b$, RH is faster, otherwise Estrin is. It is fast to check the condition at run-time.

size(a_i) = 64, size(x) = 65536 = 2^{16}



$\text{size}(a_i) = 1048576 = 2^{20}$, $\text{size}(x) = 24576$



Threshold issues (ET variant)

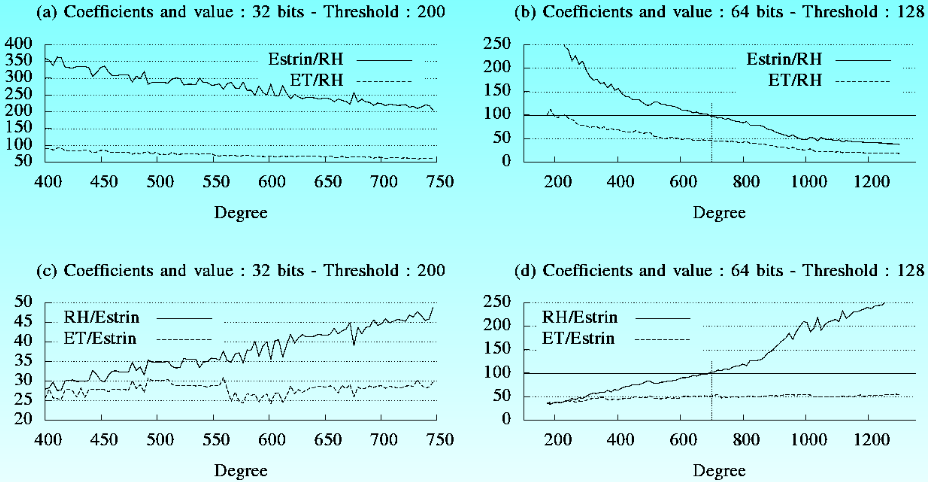
It is not always convenient for Estrin to recurse too much, in particular when n is small.

Split $p(x)$ in $p_i(x)$ according to a threshold $1 \leqslant \tau \in \mathbb{N}$, to be possibly adjusted so to have a completely balanced case.

$$p_i(x) = \sum_{j=0}^{\min\{d-i\tau,\tau-1\}} a_{i\tau+j} x^{i\tau+j} \quad ; \quad i = 0, \ldots, d' = \left\lceil \frac{d+1}{\tau} \right\rceil - 1$$

This way, $p(x) = \sum_{i=0}^{d'} p_i(x)(x^\tau)^i \Longrightarrow$ (ET variant)

1. first compute $a'_i = p_i(x)$ with Ruffini-Horner
2. then evaluate $y = x^\tau$ and $\sum_{i=0}^{d'} a'_i y^i$ with Estrin.


(a) Coefficients and value : 32 bits - Threshold : 200
Estrin/RH
ET/RH
400
350
300
250
200
150
100
50
400 450 500 550 600 650 700 750
Degree
(b) Coefficients and value : 64 bits - Threshold : 128
Estrin/RH
ET/RH
250
200
150
100
50
0
200 400 600 800 1000 1200
Degree
(c) Coefficients and value : 32 bits - Threshold : 200
RH/Estrin
ET/Estrin
50
45
40
35
30
25
20
400 450 500 550 600 650 700 750
Degree
(d) Coefficients and value : 64 bits - Threshold : 128
RH/Estrin
ET/Estrin
250
200
150
100
50
0
200 400 600 800 1000 1200
Degree

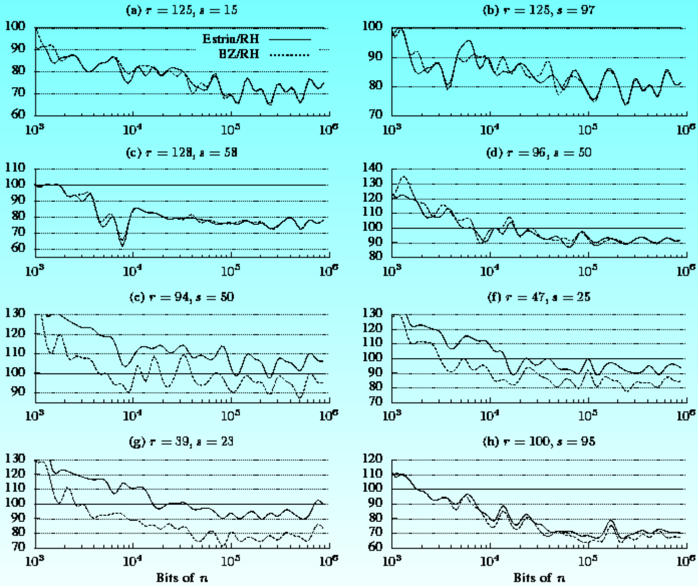
Dealing with “sparse” polynomials

Is this method general and effective?

- One may wonder if the method described here is fast also when most of the coefficients are zero.
- This is not a method specialised for sparse polynomials, but it performs pretty well anyway. For example, if we evaluate a monomial ax^d with Estrin, we basically compute all x^{2^i} and multiply the ones needed for the term x^d.
- On the next slide we show some timings obtained for binomials:

$$ax^r + bx^s$$

with the **same** code optimised for dense polynomials.


(a) r = 125, s = 15
Estrin/RH
BZ/RH
(b) r = 125, s = 97
(c) r = 128, s = 58
(d) r = 96, s = 50
(e) r = 94, s = 50
(f) r = 47, s = 25
(g) r = 39, s = 23
(h) r = 100, s = 95
Bits of n
Bits of n

Conclusions

- We have shown that Estrin paradigm applied recursively or iteratively is very effective for polynomial evaluation in long integers; moreover, faster new variants have been presented, obtaining asymptotically better algorithms solving such a basic problem in algebra.
- Although the paper focus is on integers, the strategy described in this work can be more widely used, e.g. when the coefficients and the value are fractions, or polynomials. For this latter case (polynomial composition) a similar approach can also be found in (Hart, Novocin '11 - to appear).
- In general, the straightforward Estrin's scheme, and possibly the ET and BZ variants, should be considered every time a polynomial evaluation involves values with powers that grow in size with the growing exponent and asymptotically fast multiplication algorithms are available.

That’s all folks !

Thank you very much for your very kind attention

Questions ?

Presentation will be available on the web:
http://bodrato.it/papers/#SYNASC2011,