Making change for 10^{20}

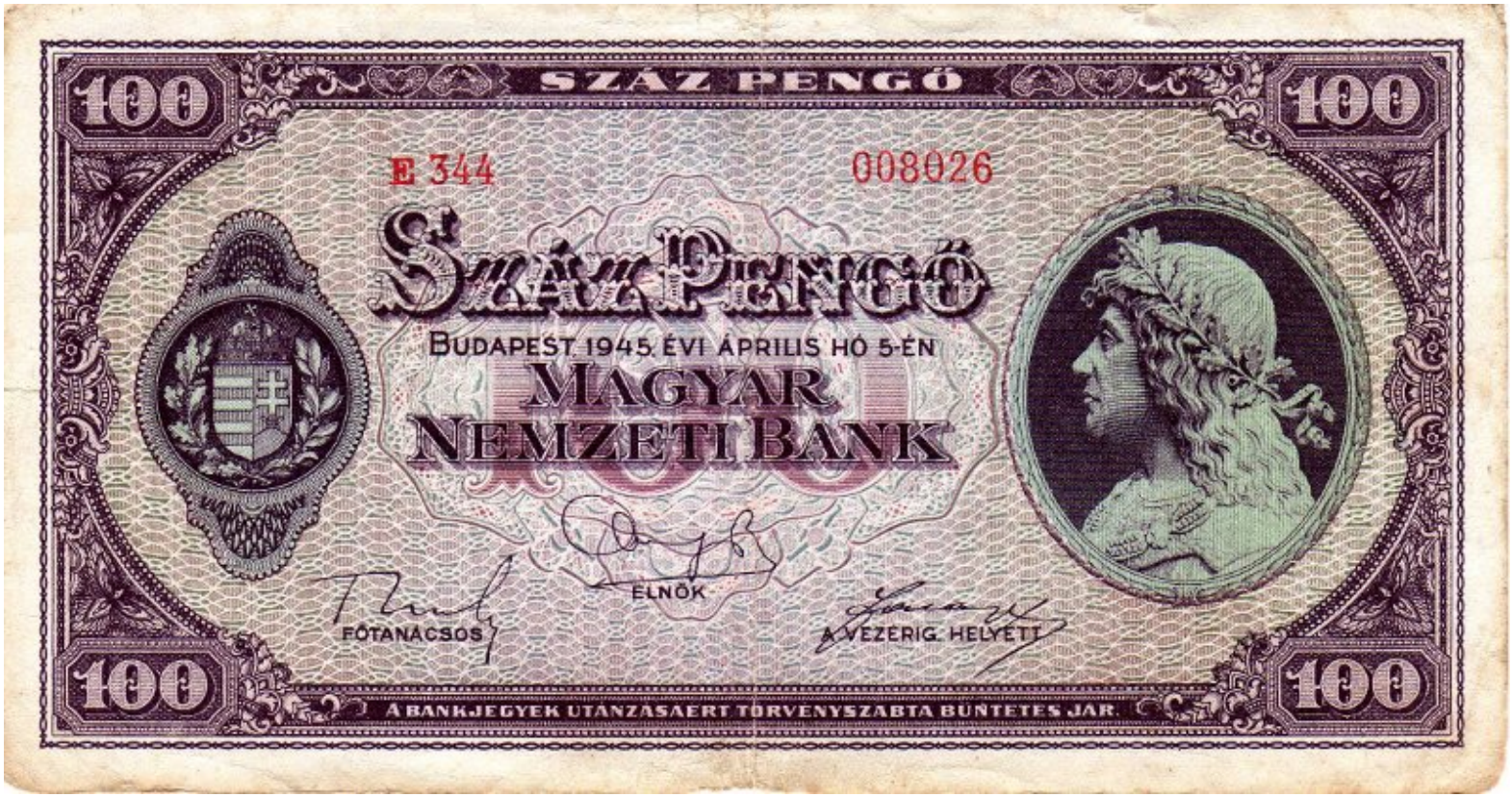
Fredrik Johansson, RISC

May 22, 2014

TU Kaiserslautern

Hungarian pengő (1 P, 1926)

10^2 pengő (April 1945)



100 pengő

10^3 pengő (July 1945)

1000 pengő

10^4 pengő (July 1945)

10,000 pengő

10^5 pengő (October 1945)

100,000 pengő

10^6 pengő (November 1945)

1,000,000 pengő

10^7 pengő (November 1945)

10,000,000 pengő

10^8 pengő (March 1946)

100,000,000 pengő

10^9 pengő (March 1946)

1,000,000,000 pengő

10^{10} pengő (April 1946)

10,000 milpengő = 10,000,000,000 pengő

10^{11} pengő (April 1946)

100,000 milpengő = 100,000,000,000 pengő

10^{12} pengő (May 1946)

1,000,000 milpengő = 1,000,000,000,000 pengő

10^{13} pengő (May 1946)

10,000,000 milpengő = 10,000,000,000,000 pengő

10^{14} pengő (June 1946)

100,000,000 milpengő = 100,000,000,000,000 pengő

10^{15} pengő (June 1946)

1,000,000,000 milpengő = 1,000,000,000,000,000 pengő

10^{16} pengő (June 1946)

10,000 b.pengő = 10,000,000,000,000,000 pengő

10^{17} pengő (June 1946)

100,000 b.pengő = 100,000,000,000,000,000 pengő

10^{18} pengő (June 1946)

1 million b.pengő = 1,000,000,000,000,000,000 (1 quintillion) pengő

10^{19} pengő (June 1946)

10 million b.pengő $=$ 10,000,000,000,000,000,000 (10 quintillion) pengő

10^{20} pengő (June 1946)

100 million b.pengő = 100,000,000,000,000,000,000
(100 quintillion) pengő

August 1946

Making change (partitions)

$$10^{20} = 1 + 3 + 3 + 99999999999999999993$$

Number of partitions: $p(10^{20})$

Recent result

Theorem (FJ+Popeye, 2014)

There are exactly

$$1838176508344882\ldots231756788091448$$

(11,140,086,260 digits) different ways to make change for a 10^{20}*-pengő banknote using stacks of 1-pengő coins.*

Growth of $p(n)$

$$p(n) \sim \frac{1}{4n\sqrt{3}}\, e^{\pi\sqrt{2n/3}}$$

$p(n)$ has $\sim n^{1/2}$ digits

$p(10) = 42$
$p(100) = 190569292$
$p(1000) \approx 2.4 \times 10^{31}$
$p(10000) \approx 3.6 \times 10^{106}$
$p(100000) \approx 2.7 \times 10^{346}$
$p(1000000) \approx 1.5 \times 10^{1107}$

Euler's method (1748) to compute $p(n)$

$$\sum_{n=0}^{\infty} p(n)x^n = \prod_{k=1}^{\infty} \frac{1}{1-x^k} = \left(\sum_{k=-\infty}^{\infty} (-1)^k x^{k(3k-1)/2} \right)^{-1}$$

$$p(n) = \sum_{k=1}^{n} (-1)^{k+1} \left(p(n - \tfrac{k(3k-1)}{2}) + p(n - \tfrac{k(3k+1)}{2}) \right)$$

Complexity: $O(n^{3/2})$ integer operations, $O(n^2)$ bit operations

Quasi-optimal vector computation

Use fast power series arithmetic to expand

$$\frac{1}{f(x)} = p(0) + p(1)x + \ldots + p(n)x^n + O(x^{n+1})$$

Computing $p(0), \ldots, p(n)$ **simultaneously**:

- $O(n^{3/2+o(1)})$ bit operations in $\mathbb{Z}$
- $O(n^{1+o(1)})$ bit operations in $\mathbb{Z}/m\mathbb{Z}$ for fixed m

Calkin et al (2007) computed $p(n)$ mod m for all $n \leq 10^9$ and primes $m \leq 103$

Quasi-optimal computation of $p(n)$

Theorem (FJ, 2011)

The isolated value $p(n)$ can be computed in time $O(n^{1/2} \log^{4+o(1)} n)$.

- Note that $p(n)$ is not holonomic
- No analogous algorithm e.g. for Bell numbers (set partitions)

The HRR formula

$$p(n) = \sum_{k=1}^{\infty} \frac{\sqrt{k}\, A_k(n)}{\pi\sqrt{2}} \frac{d}{dn} \left(\frac{\sinh\left[\frac{\pi}{k}\sqrt{\frac{2}{3}\left(n - \frac{1}{24}\right)}\right]}{\sqrt{n - \frac{1}{24}}} \right)$$

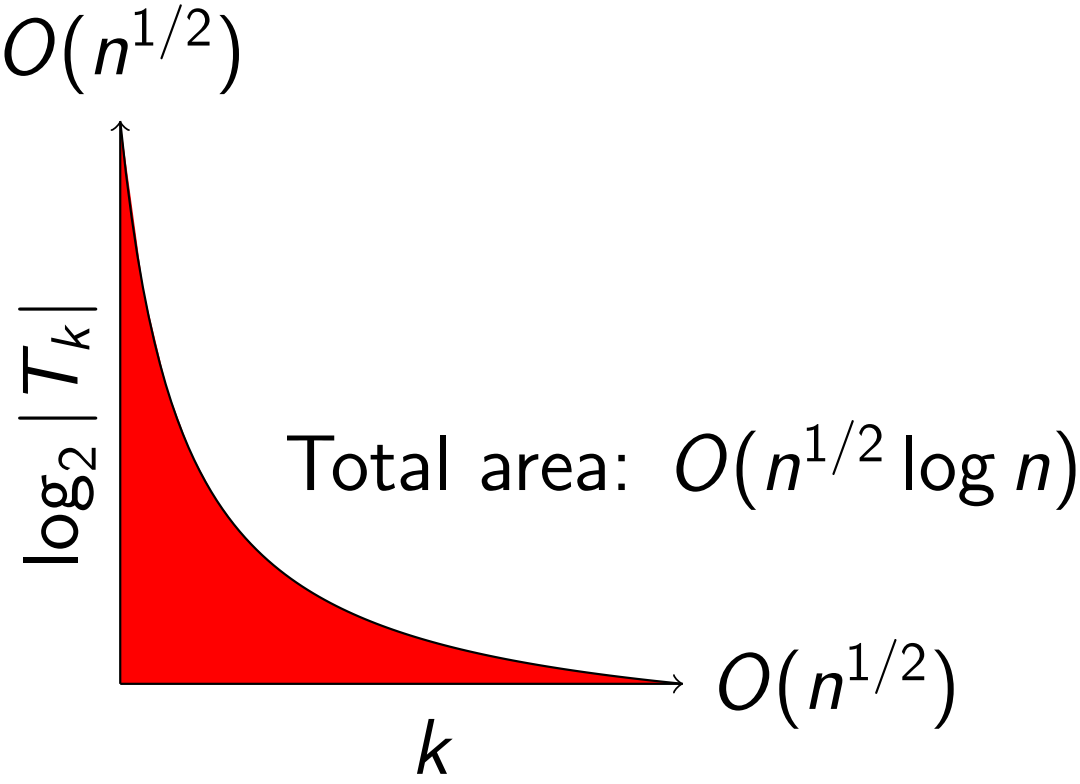
$$A_k(n) = \sum_{\substack{0 \le h < k \\ \gcd(h,k)=1}} e^{\pi i \left[s(h,k) - \frac{1}{k}2nh\right]}$$

$$s(h,k) = \sum_{i=1}^{k-1} \frac{i}{k} \left(\frac{hi}{k} - \left\lfloor \frac{hi}{k} \right\rfloor - \frac{1}{2} \right)$$

Hardy and Ramanujan (1917), Rademacher (1936)

Cost of numerical evaluation

$$p(n) \approx \sum_{k=0}^{n^{1/2}} T_k, \quad \log_2 |T_k| = O(n^{1/2}/k)$$



Need to approximate T_k in quasi-optimal time.

Evaluating exponential sums

$$A_k(n) = \sum_{\substack{0 \le h < k \\ \gcd(h,k)=1}} e^{\pi i \left[s(h,k) - \frac{1}{k} 2nh \right]}$$

$$s(h,k) = \sum_{i=1}^{k-1} \frac{i}{k} \left(\frac{hi}{k} - \left\lfloor \frac{hi}{k} \right\rfloor - \frac{1}{2} \right)$$

Naively:

- $O(k^2)$ arithmetic operations for $A_k(n)$
- $O(n^{3/2})$ arithmetic operations for $p(n)$

Fast computation of Dedekind sums

Let $0 < h < k$ and let $k = r_0, r_1, \ldots, r_{m+1} = 1$ be the sequence of remainders in the Euclidean algorithm for $\gcd(h, k)$. Then

$$s(h,k) = \frac{(-1)^{m+1} - 1}{8} + \frac{1}{12} \sum_{j=1}^{m+1} (-1)^{j+1} \frac{r_j^2 + r_{j-1}^2 + 1}{r_j r_{j-1}}.$$

- $O(\log k)$ integer ops to evaluate $s(h, k)$
- $O(k \log k)$ integer ops to evaluate $A_k(n)$
- $O(n \log n)$ integer ops to evaluate $p(n)$

Still not good enough!

$A_k(n)$ using prime factorization of k

Whiteman (1956):

$$A_{p_1^{e_1}\cdots p_i^{e_i}}(n) = \sqrt{\frac{s}{t}} \cos\left(\frac{\pi r_1}{24k_1}\right) \cdots \cos\left(\frac{\pi r_i}{24k_i}\right)$$

Requires solving quadratic equations modulo divisors of k (careful bit complexity analysis).

Only $O(\log k)$ cosines, so the numerical evaluation becomes fast enough!

Software

FLINT (`http://flintlib.org/`)

Arb (`http://fredrikj.net/arb/`)

- Numbers, polynomials matrices over $\mathbb{R}$, $\mathbb{C}$
- Ball arithmetic (rigorous error bounds)
- Asymptotically fast algorithms

2011 implementation of $p(n)$

- 500 times faster than Mathematica
- $p(10^k)$ computed up to $k = 19$
- 22 billion congruences $p(Ak + B) \equiv 0 \bmod m$ (Weaver's algorithm)

$$p(711647853449k + 485138482133) \equiv 0 \bmod 13$$

$$p(289952442924860052459470 69k + 289952213369764311353210 47) \equiv 0 \bmod 29$$

2014 implementation (in Arb)

- Rigorous error bounds
- 3× faster
- 3× less memory

Time	Memory	2011	2014
10 hours	15 GB	$p(10^{17})$	$p(10^{18})$
30 hours	50 GB	$p(10^{18})$	$p(10^{19})$
100 hours	150 GB	$p(10^{19})$	$p(10^{20})$

Sources of improvement

- Parallel computation (two threads)
- Faster exponential function
- Faster roots of unity
- Fewer terms in HRR series
- Faster FFT multiplication (Bill Hart)
- Slightly faster hardware

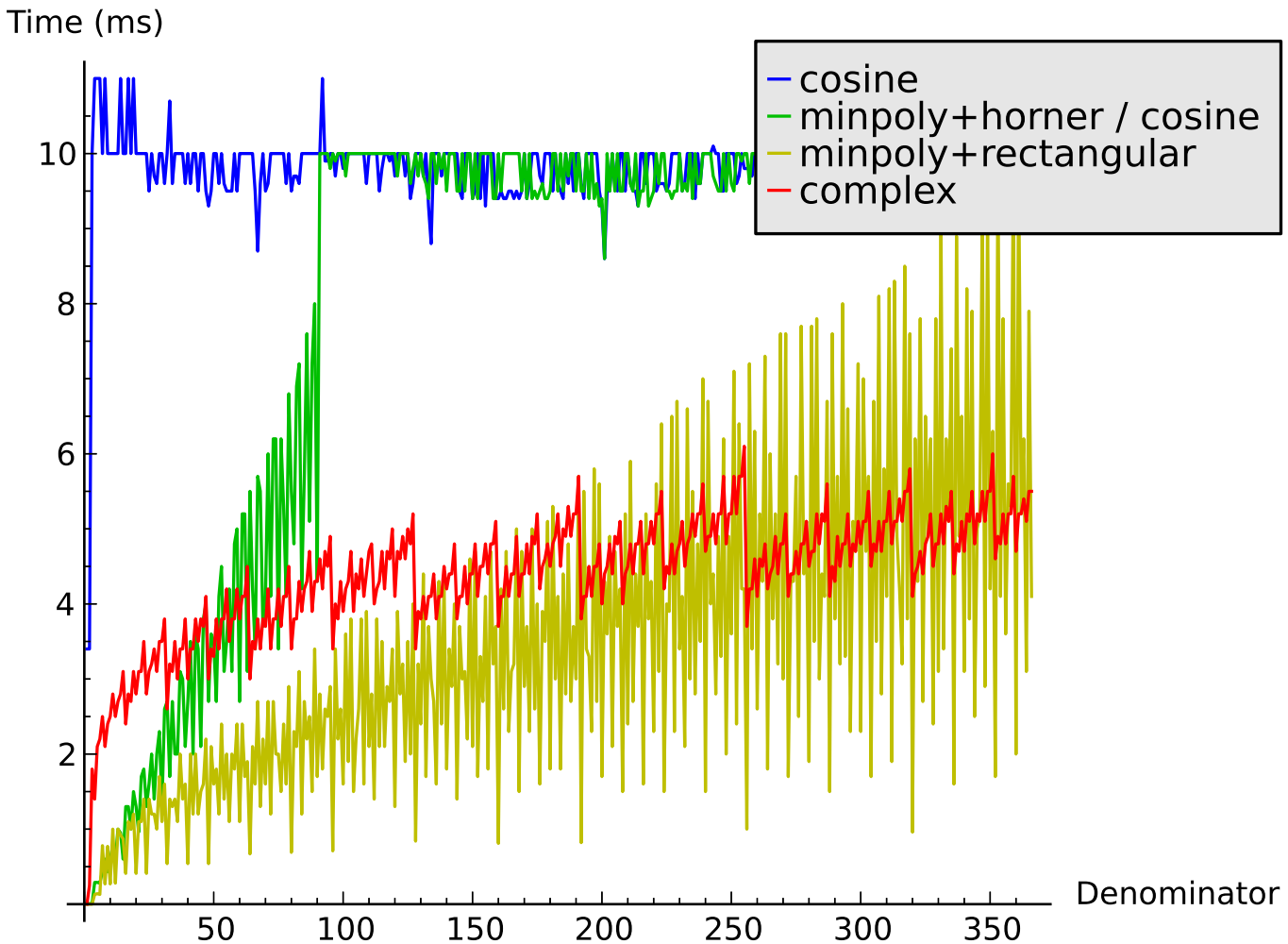
Fast roots of unity

Low precision: evaluate $\cos(p\pi/q)$ (MPFR)

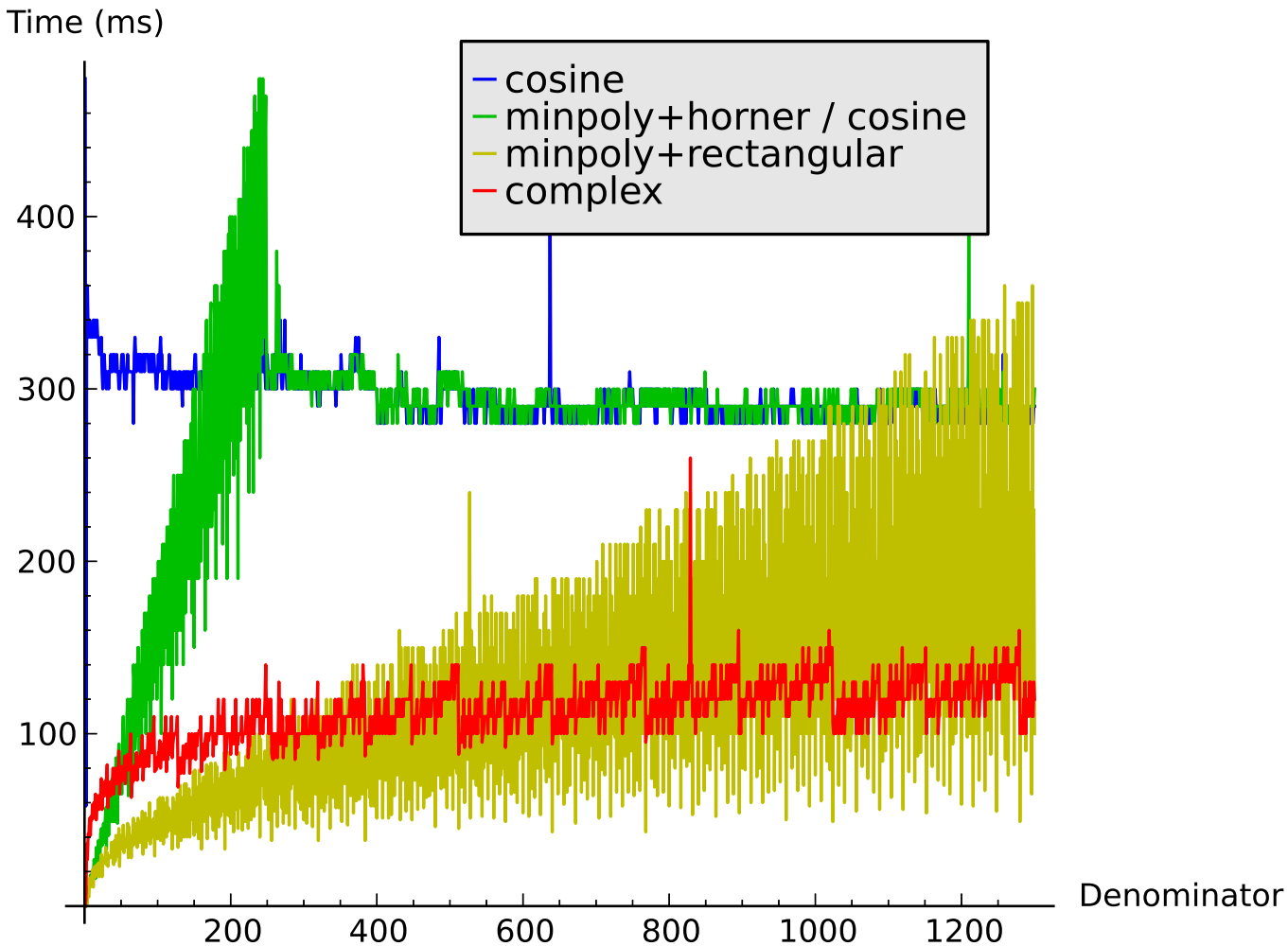
High precision: Newton iteration

- Real: minpoly of $\cos(p\pi/q)$
- Complex: $x^q \pm 1$

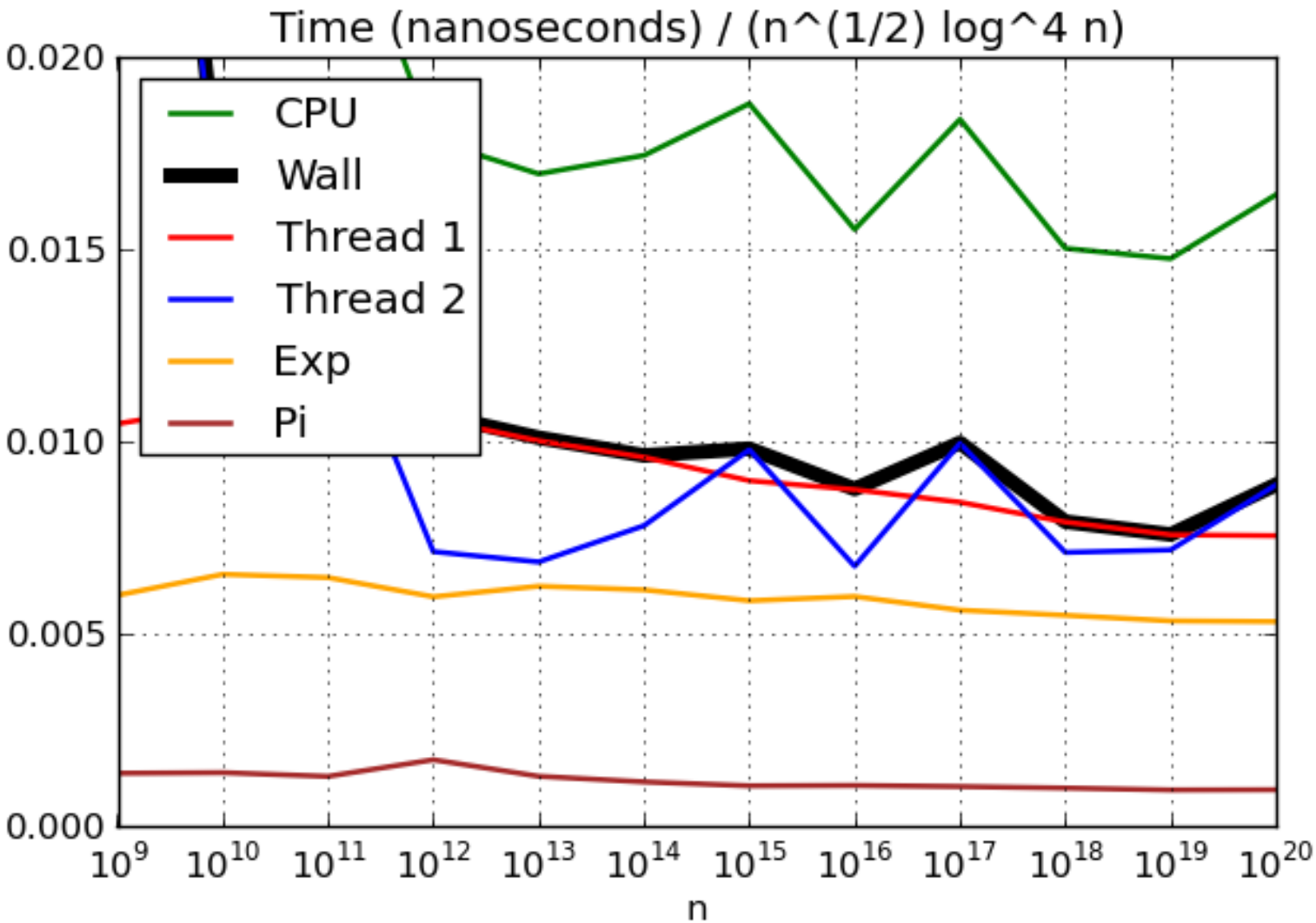
Roots of unity (10^4 digits)



Roots of unity (10^5 digits)



Time breakdown for $p(n)$



In numbers

n	Mem	Pi	Exp	T1	T2	Wall	CPU
10^{17}	4.5	0.2	1.2	1.7	2.1	2.1	3.8
10^{18}	13	0.8	4.5	6.5	5.8	6.5	12
10^{19}	38	3	17	24	23	24	47
10^{20}	128	12	66	94	111	111	205

Mem: GB, timings: hours

Images and history:
http://en.wikipedia.org/wiki/Hungarian_pengo