

APPROXIMATELY COUNTING SEMISMOOTH INTEGERS

ERIC BACH AND JONATHAN SORENSON

ABSTRACT. An integer n is (y, z) -semismooth if $n = pm$ where m is an integer with all prime divisors $\leq y$ and p is 1 or a prime $\leq z$. Large quantities of semismooth integers are utilized in modern integer factoring algorithms, such as the number field sieve, that incorporate the so-called *large prime* variant. Thus, it is useful for factoring practitioners to be able to estimate the value of $\Psi(x, y, z)$, the number of (y, z) -semismooth integers up to x , so that they can better set algorithm parameters and minimize running times, which could be weeks or months on a cluster supercomputer. In this paper, we explore several algorithms to approximate $\Psi(x, y, z)$ using a generalization of Buchstab's identity with numeric integration.

1. INTRODUCTION

The security of the public-key cryptosystem RSA [18, 22] is based on the practical difficulty of integer factoring.

The fastest current general-purpose integer factoring algorithm is the number field sieve [9, 17], which in its basic form makes use of *smooth numbers*, integers with only small prime divisors. This has inspired research into algorithms to approximately count smooth numbers [5, 6, 15, 20, 24, 26, 27]. However, most implementations of the number field sieve make use of the so-called *large prime* variant [9, §6.1.4]. So we want, in fact, to count smooth integers that admit at most one slightly larger prime divisor, or *semismooth* numbers. (See, for example, the details on the factorization of a 768-bit RSA modulus [16] where smoothness bounds are discussed near the end of §2.2.)

The principal contribution of this paper is twofold:

- (1) We present data showing that the key to estimating $\Psi(x, y, z)$ accurately is an algorithm to estimate $\Psi(x, y)$ accurately, and
- (2) We present head-to-head comparisons of five algorithms for estimating $\Psi(x, y, z)$.

Previous work was done by Bach and Peralta [3] and generalized by Zhang and Ekkelkamp [10, 29]; we discuss this below.

This paper is organized as follows. We begin with some definitions, and briefly discuss computing exact counts of semismooth integers. We then give our main theoretical result, a generalized Buchstab identity, which together with numerical integration, is the basis of all our algorithms. We then present five different algorithms in some detail, two based on the Dickman ρ function and three based on the saddle point methods of Hildebrand and Tenenbaum [14], along with empirical results for each algorithm. As one might expect, we discover a tradeoff in algorithm

TABLE 1. Exact Values of $\Psi(x, y, z)$, $x = 2^{40}$

y	$z = 2^{10}$	$z = 2^{12}$	$z = 2^{14}$	$z = 2^{16}$	$z = 2^{18}$	$z = 2^{20}$
2^2	58916	170906	503392	1500366	4513650	13597105
2^4	6132454	15111450	36766896	88920834	213965871	508848834
2^6	323105012	678707129	1326493628	2499496319	4603776946	8298713253
2^8	3157707079	6694272918	11837179134	19296840890	30059136386	45290571262
2^{10}	7138986245	21494669620	39400743040	61719198990	89501569374	123782024151
2^{12}	—	30641713551	68600140477	111769092210	160884758713	215725604647
2^{14}	—	—	80324574755	145583683889	214469637137	286977146180
2^{16}	—	—	—	155283653287	241316058768	329068435579
2^{18}	—	—	—	—	248857736183	349745847766
2^{20}	—	—	—	—	—	354983289990

choice between speed and accuracy. We follow this up with an elaboration on some numerical details.

2. DEFINITIONS

Let $P(n)$ denote the largest prime divisor of the positive integer n , with $P(1) = 1$. An integer n is *y-smooth* if $P(n) \leq y$, and $\Psi(x, y)$ counts the integers $n \leq x$ that are *y-smooth*.

An integer n is *(y, z)-semismooth* if we can write $n = mp$ where m is *y-smooth* and $p \leq z$ is a prime or 1. $\Psi(x, y, z)$ counts the integers $n \leq x$ that are *(y, z)-semismooth*. (Generalizations to more than one exceptional prime have been defined by Zhang and Eckelkamp [10, 29].) Observe that $\Psi(x, y, y) = \Psi(x, y)$, the function $\Psi(x, y, x)$ counts integers whose second-largest prime divisor is bounded by y , and $\Psi(x, 1, z) = \min\{\pi(x), \pi(z)\}$, where $\pi(x)$ is the number of primes up to x .

Our basic unit of work is the floating point operation. Along with the four basic arithmetic operations $(+, -, \times, \div)$ we include square roots, logarithms, and exponentials, since their complexity is close to that of multiplication (see for example [7]).

3. EXACT COUNTS

Using a prime number sieve, such as the sieve of Eratosthenes, we can completely factor all integers up to x in $O(x \log \log x)$ arithmetic operations, and thereby compute exact values of $\Psi(x, y, z)$. Of course this is not a practical approach for large x , but it is useful for evaluating the accuracy of approximation algorithms, which is what we do here. So we wrote a program to do this, based on a segmented sieve of Eratosthenes (see [25] for prime number sieve references), and we ran our program up to $x = 1099511627776 = 2^{40}$. Our results for this largest value for x appear in Table 1, which took just over 100 CPU hours to compute.

TABLE 2. $x \cdot \sigma(u, v) / \Psi(x, y, z)$, $x = 2^{40}$

y	$z = 2^{10}$	$z = 2^{12}$	$z = 2^{14}$	$z = 2^{16}$	$z = 2^{18}$	$z = 2^{20}$
2^2	7.0984e-14	1.2946e-12	2.1963e-11	3.4242e-10	4.8536e-09	6.2227e-08
2^4	0.0042182	0.007326	0.012701	0.021657	0.035951	0.058
2^6	0.26627	0.2956	0.32972	0.36872	0.41167	0.4585
2^8	0.64392	0.66898	0.68836	0.70731	0.72686	0.74754
2^{10}	0.75636	0.80963	0.82644	0.83778	0.84679	0.85628
2^{12}	—	0.84863	0.87886	0.89096	0.89979	0.90729
2^{14}	—	—	0.89495	0.9169	0.92614	0.93196
2^{16}	—	—	—	0.92275	0.93539	0.9421
2^{18}	—	—	—	—	0.93769	0.94722
2^{20}	—	—	—	—	—	0.95043

TABLE 3. $E(x, y, z) / \Psi(x, y, z)$, $x = 2^{40}$

y	$z = 2^{10}$	$z = 2^{12}$	$z = 2^{14}$	$z = 2^{16}$	$z = 2^{18}$	$z = 2^{20}$
2^2	1.6195e-13	2.9202e-12	4.8929e-11	7.5256e-10	1.0508e-08	1.325e-07
2^4	0.0063754	0.010979	0.018858	0.031834	0.052265	0.0833
2^6	0.34328	0.37895	0.42001	0.46628	0.51656	0.57018
2^8	0.76655	0.79204	0.81087	0.82903	0.84748	0.86587
2^{10}	0.87052	0.91765	0.93094	0.93899	0.94491	0.95185
2^{12}	—	0.94459	0.96506	0.97334	0.97836	0.98148
2^{14}	—	—	0.97446	0.98537	0.98863	0.99038
2^{16}	—	—	—	0.98694	0.99092	0.99305
2^{18}	—	—	—	—	0.99154	0.99516
2^{20}	—	—	—	—	—	0.99766

4. A GENERALIZED BUCHSTAB IDENTITY

We have the following version of Buchstab's identity (see for example [28, p. 365]):

$$(1) \quad \Psi(x, y) = \Psi(x, 2) + \sum_{2 < p \leq y} \Psi(x/p, p),$$

which is obtained by summing over the largest prime divisor of y -smooth integers $n \leq x$. Using this same idea gives us the following:

$$(2) \quad \Psi(x, y, z) = \Psi(x, y) + \sum_{y < p \leq z} \Psi(x/p, y).$$

As one can see, the identity is obtained by summing over the largest prime divisor.

5. APPROXIMATE COUNTS

As mentioned in the Introduction, there are many algorithms to estimate values of $\Psi(x, y)$. We could choose one of them, compute a list of primes up to z , and then apply (2) to approximate $\Psi(x, y, z)$. We found that this does, in fact, give fairly accurate estimates, but the resulting algorithms are quite slow since roughly $O(z/\log z)$ evaluations of $\Psi(x/p, y)$ (one for each p , $y < p \leq z$) are needed.

Our approach, then, is to replace the sum in (2) with an integral, and then use numeric integration to evaluate it [8, §7.2]; we used Simpson's rule. We found that in practice, the relative error introduced by replacing the sum with an integral that was then estimated, was less than the relative error introduced by the approximation algorithms for $\Psi(x, y)$.

Let us define $\text{li}(x) := \int_2^x dt/\log t$, and let $e(x) := \pi(x) - \text{li}(x)$. By the prime number theorem, $e(x) = x/\exp[\Omega(\sqrt{\log x})]$; if we assume the Riemann Hypothesis, $e(x) = O(\sqrt{x} \log x)$ [23].

We have the following (see [4, §2.7]):

Lemma 1. *Let f be a continuously differentiable function on an open interval containing $[2, z]$, and let $2 \leq y \leq z$. Then*

$$\begin{aligned} \sum_{y < p \leq z} f(p) &= \int_y^z \frac{f(t)}{\log t} dt \\ &\quad + f(z)e(z) - f(y)e(y) - \int_y^z e(t)f'(t)dt. \end{aligned}$$

Of course we cannot apply this lemma to $\Psi(x, y)$ directly, so we use an estimate instead. Define $\rho(u)$ as the unique continuous solution to

$$\begin{aligned} \rho(u) &= 1 & (0 \leq u \leq 1) \\ \rho(u-1) + u\rho'(u) &= 0 & (u > 1). \end{aligned}$$

Note that $\rho \in C^1$ for $u > 1$. Hildebrand [12] proved that for $\epsilon > 0$ we have

$$(3) \quad \Psi(x, y) = x\rho(u) \left(1 + O_\epsilon \left(\frac{\log(u+1)}{\log y} \right) \right),$$

uniformly on the set defined by $1 \leq u \leq \exp((\log y)^{3/5-\epsilon})$ and $y \geq 2$. Here, $u := u(x, y) = \log x / \log y$.

Theorem 1. *Given $2 \leq y \leq z \leq x$, let $\epsilon > 0$, and assume that $1 \leq \log(x/z)/\log y \leq \log x / \log y \leq \exp((\log y)^{3/5-\epsilon})$. Then*

$$(4) \quad \Psi(x, y, z) = \left(\Psi(x, y) + \int_y^z \frac{\Psi(x/t, y)}{\log t} dt \right) (1 + o(1)).$$

For asymptotic notation, we are assuming y is large.

Proof. Define $f(t) := (x/t)\rho(\log(x/t)/\log y)$. By (3) we have $f(p) = \Psi(x/p, y)(1 + o(1))$ for all primes $y < p \leq z$. f is differentiable and continuous, with $f'(t) \sim -f(t)/t$ (for large y). It is then straightforward to show that $e(t)f'(t) = o(f(t)/\log t)$. Also since f is decreasing we can show $e(z)f(z) - e(y)f(y) = o(\pi(z) - \pi(y))f(z) + O(|e(y)|f(y))$ and then $(\pi(z) - \pi(y))f(z) \leq \sum_{y < p \leq z} f(p)$ and $|e(y)|f(y) = o(\Psi(x, y))$. In the case when u is large, we make use of Lemma 8.1 and (61) from [28, §5.4]. We then apply Lemma 1 and use (2), and finally substitute $\Psi(x/t, y)$ back in for $f(t)$ to complete the proof. \square

It would be nice to have some function $g(y, z)$ where

$$(5) \quad \Psi(x, y, z) \approx \Psi(x, y) \cdot g(y, z)$$

TABLE 4. $HT(x, y, z)/\Psi(x, y, z)$, $x = 2^{40}$

y	$z = 2^{10}$	$z = 2^{12}$	$z = 2^{14}$	$z = 2^{16}$	$z = 2^{18}$	$z = 2^{20}$
2^2	1.0969	1.0712	1.0641	1.067	1.0728	1.0833
2^4	1.0592	1.039	1.031	1.0353	1.0507	1.0829
2^6	1.0414	1.0281	1.0205	1.024	1.0566	1.1582
2^8	1.0194	1.0165	1.0135	1.0127	1.0296	1.1338
2^{10}	1.0024	1.0104	1.0101	1.01	1.0112	1.0437
2^{12}	—	1.0043	1.0104	1.01	1.0078	1.0107
2^{14}	—	—	1.0046	1.0084	1.0115	1.0143
2^{16}	—	—	—	1.0101	1.016	1.017
2^{18}	—	—	—	—	1.0142	1.0116
2^{20}	—	—	—	—	—	1.0083

if this is possible. Rewriting (2) we have

$$(6) \quad \Psi(x, y, z) = \Psi(x, y) \cdot \left(1 + \sum_{y < p \leq z} \frac{\Psi(x/p, y)}{\Psi(x, y)} \right).$$

A very crude estimate of $\Psi(x/p, y)/\Psi(x, y) \approx 1/p$ leads to

$$(7) \quad \Psi(x, y, z) \approx \Psi(x, y) \cdot (1 + \log(\log z / \log y)).$$

In practice, this is too crude to be useful. However, this estimate can certainly be improved using, for example, Theorem 11 from [28, §5.5]. This is a possible direction for future work.

5.1. The Method of Bach and Peralta. The first algorithm to try would be to use the estimate $\Psi(x, y) \approx x\rho(u)$ from (3) and plug it into Theorem 1. This, in fact, is simply another way to derive the algorithm of Bach and Peralta [3]. They define $u := \log x / \log y$, $v := \log x / \log z$ and

$$(8) \quad \sigma(u, v) := \rho(u) + \int_v^u (\rho(u - u/w)/w)dw.$$

They then prove that for fixed u, v and $x \rightarrow \infty$,

$$(9) \quad \Psi(x, y, z) \approx x\sigma(u, v).$$

We can use (2) to obtain the same approximation, as follows:

$$\begin{aligned}
\Psi(x, y, z) &\approx \Psi(x, y) + \int_y^z (\Psi(x/t, y)/\log t)dt \\
&\approx x \cdot \rho(\log x / \log y) \\
&\quad + \int_y^z (x/t \log t) \rho(\log(x/t) / \log y) dt \\
&= x \cdot \left(\rho(u) + \int_v^u (\rho(u - u/w)/w)dw \right) \\
&= x \cdot \sigma(u, v).
\end{aligned}$$

We adapted code written by Peralta to compute values of the Dickman ρ function (accurate to roughly 8 decimal digits) to compute σ . Bach and Peralta discuss methods to compute both ρ and σ in some detail in [3]. We present the results

from this algorithm in Table 2. As for all our algorithms, we show the ratio of what the algorithm produces as an estimate divided by the actual values we computed earlier. The closer we are to 1 the better the estimates.

This algorithm is fast, but the results are not as good as we might desire. As one might expect, they are about as good as what was found for estimating counts of smooth numbers using (3) in [15].

5.2. Ekkelkamp's Improvement. Ekkelkamp [10] pointed out that $\sigma(u, v)$ could be made more accurate by adding a quantity which, in our notation, is

$$\frac{(1-\gamma)x}{\log x} \left[\rho(u-1) + \int_v^u \frac{\rho(u-u/w-1)}{w-1} dw \right].$$

This can be derived by using the better approximation

$$\Psi(x, y) \approx x\rho(u) + \frac{(1-\gamma)x}{\log x} \rho(u-1),$$

due to Ramaswami [21, Theorem 1]. To get her formula, substitute $w = 1/\lambda$ in the integral; note that we need the first term inside the brackets, since our definition of semismoothness differs from hers. (Our “large prime” can be 1.) This correction does not require much additional effort; essentially just one more numerical integration. Let $E(x, y, z)$ denote this approximation.

Our experiments with this indicate that the additional term enhances accuracy significantly when y and z are large. This is roughly the bottom corner of Table 2. However, the accuracy still drops off dramatically for smaller values of y . See Table 3.

5.3. A Saddlepoint Method. Our third algorithm is based on Algorithm HT for estimating $\Psi(x, y)$ presented in [15]. Define

$$\begin{aligned} \zeta(s, y) &:= \prod_{p \leq y} (1 - p^{-s})^{-1}; \\ \phi(s, y) &:= \log \zeta(s, y); \\ \phi_k(s, y) &:= \frac{d^k}{ds^k} \phi(s, y) \quad (k \geq 1). \end{aligned}$$

The functions ϕ_k can be expressed as sums over primes. Indeed, we have

$$\begin{aligned} \phi(s, y) &= - \sum_{p \leq y} \log(1 - p^{-s}); \\ \phi_1(s, y) &= - \sum_{p \leq y} \frac{\log p}{p^s - 1}; \\ \phi_2(s, y) &= \sum_{p \leq y} \frac{p^s (\log p)^2}{(p^s - 1)^2}. \end{aligned}$$

Thus, with a list of primes up to y , the quantities $\zeta(s, y)$, $\phi_1(s, y)$, and $\phi_2(s, y)$ can be computed in $O(y/\log y)$ floating point operations.

Define

$$HT(x, y, s) := \frac{x^s \zeta(s, y)}{s \sqrt{2\pi \phi_2(s, y)}},$$

TABLE 5. $HT_f(x, y, z)/\Psi(x, y, z)$, $x = 2^{40}$

y	$z = 2^{10}$	$z = 2^{12}$	$z = 2^{14}$	$z = 2^{16}$	$z = 2^{18}$	$z = 2^{20}$
2^2	1.0969	1.0712	1.0641	1.067	1.0728	1.0833
2^4	1.0592	1.039	1.031	1.0353	1.0507	1.0829
2^6	1.0414	1.0281	1.0205	1.024	1.0566	1.1582
2^8	1.0194	1.0165	1.0135	1.0127	1.0296	1.1338
2^{10}	0.99773	1.0069	1.007	1.0073	1.0088	1.0413
2^{12}	—	1.0152	1.0186	1.0171	1.0141	1.0164
2^{14}	—	—	1.0177	1.0186	1.0203	1.0222
2^{16}	—	—	—	1.021	1.0247	1.0246
2^{18}	—	—	—	—	1.021	1.0172
2^{20}	—	—	—	—	—	1.013

and let α be the unique solution to $\phi_1(\alpha, y) + \log x = 0$. Hildebrand and Tenenbaum proved the following [13]:

Theorem 2.

$$(10) \quad \Psi(x, y) = HT(x, y, \alpha) \cdot \left(1 + O\left(\frac{1}{u} + \frac{(\log y)}{y}\right)\right)$$

uniformly for $2 \leq y \leq x$.

This gives us Algorithm HT [15]:

- (1) Find the primes up to y .
- (2) Compute an approximation α' to α using binary search and Newton's method. Make sure that $|\alpha - \alpha'| = O(1/(u \log x))$.
- (3) Output $HT(x, y, \alpha')$.

Write $HT(x, y)$ for the value output in the last step. The running time is $O\left(\frac{y \log \log x}{\log y} + \frac{y}{\log \log y}\right)$ floating point operations.

We simply plugged Algorithm HT into Theorem 1 to estimate $\Psi(x, y, z)$ using the saddle point method as follows:

$$(11) \quad HT(x, y, z) := HT(x, y) + \int_y^z (HT(x/t, y)/\log t) dt.$$

In Table 4 we give the results for this algorithm, which are quite good. The method is, however, a bit slow.

Using the summation algorithms described in [2], we can lower the exponent of y in the running time from 1 to $2/3$. We give some details for this in §6.3. We did not implement this improvement, however, because it would not change the computed results, and the method of the section below is faster.

5.4. Assuming Riemann's Hypothesis. This is the same as Algorithm HT, only sums over primes (ζ, ϕ_1, ϕ_2) above roughly \sqrt{y} are estimated using the prime number theorem plus the Riemann Hypothesis [24, 23]. It is *much* faster than Algorithm HT and nearly as accurate; its running time is roughly \sqrt{y} . Let $HT_f(x, y)$ denote the estimate this algorithm computes for $\Psi(x, y)$, and $HT_f(x, y, z)$ the estimate after using $HT_f(x, y)$ with Theorem 1. We present our results in Table 5.

We recommend this method.

TABLE 6. $S(x, y, z)/\Psi(x, y, z)$, $x = 2^{40}$

y	$z = 2^{10}$	$z = 2^{12}$	$z = 2^{14}$	$z = 2^{16}$	$z = 2^{18}$	$z = 2^{20}$
2^2	0	0	0	0	0	0
2^4	0	0	0	0	0	2.3032
2^6	0.57462	0.60753	0.64352	0.68488	0.7388	0.82405
2^8	0.90295	0.90963	0.91195	0.91382	0.92929	1.0201
2^{10}	0.94404	0.93337	0.92495	0.9175	0.9108	0.93201
2^{12}	—	0.93348	0.90717	0.89212	0.87883	0.87083
2^{14}	—	—	0.90403	0.87455	0.85967	0.84873
2^{16}	—	—	—	0.88064	0.85461	0.83658
2^{18}	—	—	—	—	0.85917	0.82827
2^{20}	—	—	—	—	—	0.83353

5.5. Suzuki's Algorithm. In successive papers, Suzuki [26, 27] develops a very fast algorithm, with cost $O(\sqrt{\log x \log y})$ operations, using the saddle point method to estimate $\Psi(x, y)$. This is based on good approximations for α and the prime sums ζ, ϕ_1, ϕ_2 using the prime number theorem.

For $u > 1$, let ξ be the positive solution to the equation $e^\xi = 1 + u\xi$, or equivalently, $\xi = \log(1 + u\xi)$. This last equation implies that $\xi \approx \log(u \log u)$, and can be used iteratively to evaluate ξ . (See §6.2 for more information on this point.)

Let $\gamma = 0.57721\dots$ be Euler's constant. We now define

$$(12) \quad \alpha_s := 1 - \frac{\xi}{\log y}$$

and

$$(13) \quad S(x, y) := \frac{x^{\alpha_s} e^{\gamma + \int_0^\xi t^{-1}(e^t - 1)dt}}{\alpha_s \sqrt{2\pi u(1 + (\log x)/y)}}.$$

Suzuki proves the following [27, Theorem 1.1]:

Theorem 3. *Let $\epsilon \leq 1/2$. If $(\log \log x)^{5/3-\epsilon} < \log y < e^{-1}(1 - \epsilon) \log x$, then*

$$(14) \quad \Psi(x, y) = S(x, y)(1 + o(1)).$$

Suzuki proposed using the midpoint method to evaluate the integral $\int_0^\xi t^{-1}(e^t - 1)dt$. We used its Maclaurin series $\sum_{n \geq 1} \xi^n / (n \cdot n!)$ [1, formulas 5.1.10 and 5.1.40]. We write $S(x, y, z)$ for the function to estimate $\Psi(x, y, z)$ using $S(x, y)$ to estimate $\Psi(x, y)$ in Theorem 1.

Our results are presented in Table 6. We found the algorithm to be extremely fast, but not accurate for small y , which is not surprising given the approximations used.

Earlier, in [26], Suzuki discussed $\text{HT}(x, y, \alpha_s)$ as an approximation to $\Psi(x, y)$. Although this is faster than Algorithm HT by a factor of $\log \log x$, it is not as accurate, and so we chose not to test its use.

5.6. Speed. Below we give timing results for the algorithms presented above. This is the total time it took to compute the estimates given in the tables in this section.

<i>Algorithm</i>	<i>Time in Seconds</i>
Suzuki	0.09
Bach and Peralta	0.40
Ekkelkamp	0.78
HT_f	1.90
HT	20.0

We used the Gnu `g++` compiler on a unix server with an intel CPU.

6. NUMERICAL DETAILS

We elaborate on some details from the algorithms presented above.

6.1. Estimates for α . In this subsection, we give more information about the function $\alpha(x, y)$, defined implicitly by the equation $\phi_1(\alpha, y) + \log x = 0$ (here $2 \leq y \leq x$). In particular we will prove that

$$(15) \quad \frac{1}{2 \log x} \leq \alpha \leq 2.$$

We prove the lower bound first. From the prime number sum for ϕ_1 , we see that for $s > 0$,

$$-\phi_1 \geq \frac{\log 2}{2^s - 1}.$$

So α , the solution to $\phi_1(\alpha, y) + \log x = 0$, is lower bounded by the solution β to

$$\frac{\log 2}{2^\beta - 1} = \log x.$$

Solving, we get

$$\alpha \geq \beta = \frac{\log(1 + (\log 2)/(\log x))}{\log 2} \geq \frac{1}{2 \log x},$$

the last inequality holding whenever $x \geq 2$.

Next we show the upper bound. Let ζ denote the Riemann zeta function. By examining their Dirichlet series, we can see that $-\phi_1(s, y) \leq -\zeta'/\zeta(s)$. Both sides are decreasing smooth functions of s on $(0, \infty)$. It follows that α is upper bounded by the solution to $\zeta'(s)/\zeta(s) = \log 2$, which is less than 2.

More precise information can be found in [13]. In particular,

$$\alpha = \frac{\log(1 + y/\log x)}{\log y} \left\{ 1 + O\left(\frac{\log \log(1 + y)}{\log y}\right) \right\}$$

holds uniformly for $x \geq y \geq 2$, with the explicit lower bound

$$\alpha \geq \frac{\log(1 + y/(5 \log x))}{\log y}.$$

The strength of this is similar to (15) if y is fixed and $x \rightarrow \infty$.

6.2. Computing ξ . Here we discuss some numerical methods for solving $e^\xi = 1 + u\xi$, when $u > 1$.

Let $f(x) = x - \log(1 + ux)$. Then f is convex on $(0, \infty)$ with a minimum at $x = 1 - 1/u$. This gives the lower bound $\xi \geq 1 - 1/u$. To get an upper bound, we observe that $e^x > 1 + x + x^2/2$, so ξ is no larger than the positive solution to $1 + \xi + \xi^2/2 = 1 + u\xi$, which is $2(u - 1)$. Using binary search between these bounds, we can get $\lfloor \xi \rfloor$ plus d bits of its fraction, with $O(\log u + d)$ evaluations of f .

Starting from the defining equation and taking logarithms, we get

$$\xi = \log(u\xi + 1).$$

Suzuki [26, Lemma 2.2] proves that this iteration, starting from $\log u$, is linearly convergent to ξ . (Here $u > e$ is fixed.)

In practice, we can use the Newton iteration

$$\xi := \xi - \frac{(\xi - \log(1 + u\xi))(1 + u\xi)}{1 + u(\xi - 1)},$$

starting with the upper bound $2(u - 1)$. By convexity, the iterates decrease toward the root. We tested values of u from 2 to 1000, and for these u , about 5 iterations were enough to get machine accuracy (about 15D). (Note that when x is large, $f(x) \sim x$, so even if we start with a large upper bound, the second iterate will be much closer to the root.)

Newton iteration does not work well when u is close to 1, for the following reason. Suppose that $u = 1 + \epsilon$. Then, we have $\epsilon + O(\epsilon^2) < \xi < 2\epsilon$, making $\xi - \log(1 + u\xi)$ vanish to first order in ϵ . Thus, when computing this factor, we will lose precision due to cancellation.

If special function software is available, ξ can be expressed using the Lambert W function. For example, in the notation of a well known computer algebra system of Canadian origin, $\xi = -1/u - \text{LambertW}(-1, -e^{-1/u}/u)$. (The argument -1 indicates which branch should be used.)

6.3. ATM Summation. The purpose of the next two subsections is to justify the remark made earlier that the cost of evaluating the formula HT can be lowered to $y^{2/3+o(1)}$. Here, the “ $o(1)$ ” term includes factors of order $\log x$, so we are implicitly assuming that x is not outrageously large.

If f is a function defined on the positive integers, it is *multiplicative* if $f(mn) = f(m)f(n)$. This is a stronger requirement than is usual in number theory, where m and n need only be coprime. The concept of an *additive* function is defined similarly; we require $f(mn) = f(m) + f(n)$. We will call f an *ATM function* (additive times multiplicative) if $f = gh$, where g is additive and h is multiplicative.

The paper [2] gave an algorithm that evaluates the prime sum $\sum_{p \leq y} f(p)$, with f an ATM function, in $y^{2/3+o(1)}$ steps. This generalized a previously known result, also explained in [2], in which f could be multiplicative.

We now explain how these summation algorithms can be used to evaluate ϕ_1 and $\log \zeta$. The basic idea is to approximate each of these by a “small” number of ATM or multiplicative prime sums. With $\log \zeta$ in hand, we can exponentiate to get ζ .

We first assume $s > 0$.

Let us consider ϕ_1 first. By summing geometric series, we see that

$$-\phi_1 = \sum_{p \leq y} \frac{\log p}{p^s - 1} = \sum_{k \geq 1} \sum_{p \leq y} \frac{\log p}{p^{ks}}.$$

Note that each inner sum involves an ATM function. We will restrict the outer sum to $1 \leq k \leq N$, and choose N to make the truncation error,

$$\sum_{k \geq N+1} \sum_{p \leq y} \frac{\log p}{p^{ks}}.$$

small.

If we interchange the order of summation, allow all $p \geq 2$, and sum geometric series, we can express the truncation error as

$$\sum_{p \geq 2} \frac{\log p}{p^{Ns}(p^s - 1)}.$$

Using the globally convergent Maclaurin series for p^s , we see that $1/(p^s - 1) \leq 1/(s \log p)$. If we plug this in, the $\log p$ factors cancel and we get the upper bound

$$\frac{1}{s} \sum_{p \geq 2} \frac{1}{p^{Ns}} \leq \frac{1}{s} \left(\frac{1}{2^{Ns}} + \int_2^\infty \frac{dt}{t^{Ns}} \right) \leq \frac{3}{s 2^{Ns}},$$

provided that $Ns \geq 2$. For us, $s \geq 1/(2 \log x)$, and with this additional assumption we get

$$[\text{truncation error}] \leq \frac{6 \log x}{2^{Ns}}.$$

Thus, to achieve truncation error less than 2^{-d} , we can use $N = \Theta((\log x)(\log \log x + d))$.

Similarly, we can use

$$\log \zeta(s, y) = - \sum_{p \leq y} \log(1 - p^{-s}) = \sum_{k \geq 1} \frac{1}{k} \sum_{p \leq y} \frac{1}{p^{ks}}.$$

Now each inner sum involves a multiplicative function. If we use only the inner sums with $k \leq N$, a similar analysis shows that choosing $N = \Theta((\log x)(\log \log x + d))$ will keep the truncation error below 2^{-d} .

6.4. Numerical Differentiation. In this subsection, we explain how to evaluate

$$\phi_2(s, y) = \sum_{p \leq y} \frac{\log^2 p \cdot p^s}{(p^s - 1)^2},$$

for use in Algorithm HT. There is no obvious way to reduce this to the kind of sums treated in [2], so we will approximate it by a difference.

Using balanced numerical differentiation [8, p. 297], we have

$$\phi_2(s, y) = \frac{\phi_1(s + h, y) - \phi_1(s - h, y)}{2h} + \epsilon, \quad \epsilon = \frac{h^2}{6} \phi_4(\eta, y)$$

for some $\eta \in [s - h, s + h]$. Let us determine how much precision will be necessary to deliver d bits of $\phi_2(s, y)$ accurately, when we use this formula.

After differentiating the sum for ϕ_2 twice, we see that

$$\phi_4(s, y) = \sum_{p \leq y} \frac{p^s \log^2 p}{(p^s - 1)^2} \times \frac{(p^{2s} + 4p^s + 1) \log^2 p}{(p^s - 1)^2}.$$

Observing that $(t^2 + 4t + 1)/(t - 1)^2$ is decreasing for $t > 1$, we get the estimate

$$\frac{h^2}{6} \phi_4(s, y) \leq \frac{h^2 \log^2 y}{6} \left(\frac{2^{2s} + 4 \cdot 2^s + 1}{(2^s - 1)^2} \right) \phi_2(s, y).$$

If $0 \leq s \leq 2$, the factor in parentheses is bounded by $15/s^2$. (Numerically, anyway.) So, if we could use exact arithmetic, numerical differentiation would give us

$$[\text{relative error}] \leq \frac{5}{2} \frac{h^2 \log^2 y}{s^2} \leq 10h^2 \log^2 x \log^2 y.$$

However, we don't have exact arithmetic, so we must also analyze the loss of precision due to cancellation. For this, we use an ad hoc theory. When h is small, the number of bits lost, when using the balanced difference formula to compute $\phi'_1(s)$, is about

$$-\log_2 \left| \frac{\phi_1(s+h) - \phi_1(s-h)}{\phi_1(s)} \right|,$$

since dividing by h causes no loss of precision. Note that this is a centered version of the usual relative error formula. Since $\phi_1(s+h) - \phi_1(s-h) \sim 2h\phi'_1(s)$, we must bound the logarithmic derivative of ϕ_1 , or, what is the same thing, relate $\phi_2 = \phi'_1$ to ϕ_1 .

In our case, we require a lower bound for ϕ_2 . Then since $t/(t-1)$ is decreasing on $(1, \infty)$,

$$\begin{aligned} \phi_2(s, y) &= \sum_{p \leq y} \frac{\log p}{p^s - 1} \times \frac{p^s \log p}{p^s - 1} \\ &\geq \sum_{p \leq y} \frac{\log p}{(p^s - 1)} \log p \geq -\phi_1(s, y) \log 2. \end{aligned}$$

Therefore,

$$\phi_1(s+h, y) - \phi_1(s-h, y) \sim 2h\phi_2(s, y) \geq -\phi_1(s, y) \cdot 2h \log 2,$$

as $h \rightarrow 0$,

Let us now translate these results into practical advice. Suppose our goal is to obtain ϕ_2 to d bits of precision, in the sense of relative error. By the exact arithmetic formula, we should choose $h \leq 2^{-d/2}/(\sqrt{10} \log^2 x)$. Then we need to use $\log_2 h^{-1} + O(1)$ guard bits in our computation. Put more crudely, unless x is very large, doubling the working precision should be enough, if we select h properly.

The following example indicates that the theory above is roughly correct. Suppose we want 10 digits of ϕ_2 , for $x = 10^6$, $y = 10^3$, and $s = 1/(2 \log x) = 0.03619\dots$. Our recipe allows us to take $h = 10^{-7}$. With 17 digit arithmetic, we obtained

$$\begin{aligned} \text{numerical derivative} &= 127790.77386350000 \\ \text{summation for } \phi_2 &= 127790.77386041727 \end{aligned}$$

which agree to 11 figures.

7. CONCLUSION AND FUTURE WORK

In summary, we recommend the estimate $HT_f(x, y, z)$ of §5.4 for approximating $\Psi(x, y, z)$. We feel it gives high accuracy while retaining sufficient speed to be very practical.

For future work, we hope to generalize our results to 2 or more large primes. We also hope to further examine estimates of the form of (5).

8. ACKNOWLEDGMENTS

We want to thank the referees, whose comments helped improve this paper.

Abstract presented at the AMS-MAA Joint Mathematics Meetings, January 2012, Boston MA, and at the CMS Summer Meeting, June 2013, Halifax Nova Scotia.

Supported in part by grants from the Holcomb Awards Committee, NSF (CCF-635355), and ARO (W911NF-09-1-0439).

REFERENCES

- [1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover, 1970.
- [2] E. Bach. Sums over primes. In A. M. Ernvall-Hytönen, M. Jutila, J. Karhumäki, and A. Lepistö, editors, *Proceedings of Conference on Algorithmic Number Theory 2007*, number 46 in Turku Centre for Computer Science, pages 40–44, 2007.
- [3] E. Bach and R. Peralta. Asymptotic semismoothness probabilities. *Math. Comp.*, 65(216):1701–1715, 1996.
- [4] E. Bach and J. O. Shallit. *Algorithmic Number Theory*, volume 1. MIT Press, 1996.
- [5] D. J. Bernstein. Bounding smooth integers. In J. P. Buhler, editor, *Third International Algorithmic Number Theory Symposium*, pages 128–130, Portland, Oregon, June 1998. Springer. LNCS 1423.
- [6] D. J. Bernstein. Arbitrarily tight bounds on the distribution of smooth integers. In Bennett, Berndt, Boston, Diamond, Hildebrand, and Philipp, editors, *Proceedings of the Millennial Conference on Number Theory*, volume 1, pages 49–66. A. K. Peters, 2002.
- [7] R. P. Brent. Multiple precision zero-finding methods and the complexity of elementary function evaluation. In J. F. Traub, editor, *Analytic Computational Complexity*, pages 151–176. Academic Press, 1976.
- [8] S. D. Conte and C. W. D. Boor. *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill Higher Education, 3rd edition, 1980.
- [9] R. Crandall and C. Pomerance. *Prime Numbers, a Computational Perspective*. Springer, 2001.
- [10] W. H. Ekkelkamp. The role of semismooth numbers in factoring large numbers. In A.-M. Ernvall-Hytönen, M. Jutila, J. Karhumäki, and A. Lepistö, editors, *Proceedings of Conference on Algorithmic Number Theory 2007*, number 46 in TUCS General Publication, pages 40–44. Turku Centre for Computer Science, 2007.
- [11] A. Granville. Smooth numbers: computational number theory and beyond. In *Algorithmic number theory: lattices, number fields, curves and cryptography*, volume 44 of *Math. Sci. Res. Inst. Publ.*, pages 267–323. Cambridge Univ. Press, Cambridge, 2008.
- [12] A. Hildebrand. On the number of positive integers $\leq x$ and free of prime factors $> y$. *Journal of Number Theory*, 22:289–307, 1986.
- [13] A. Hildebrand and G. Tenenbaum. On integers free of large prime factors. *Trans. AMS*, 296(1):265–290, 1986.
- [14] A. Hildebrand and G. Tenenbaum. Integers without large prime factors. *Journal de Théorie des Nombres de Bordeaux*, 5:411–484, 1993.
- [15] S. Hunter and J. P. Sorenson. Approximating the number of integers free of large prime factors. *Mathematics of Computation*, 66(220):1729–1741, 1997.
- [16] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, P. Gaudry, P. L. Montgomery, D. A. Osvik, H. T. Riele, A. Timofeev, P. Zimmermann, and et al. Factorization of a 768-bit rsa modulus. <http://eprint.iacr.org/2010/006.pdf>, 2010.
- [17] A. K. Lenstra and H. W. L. Jr., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin and Heidelberg, Germany, 1993.
- [18] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.
- [19] P. Moree. Nicolaas Govert de Bruijn, the enchanter of friable integers. *Indag. Math.*, 2013. to appear; available from arxiv.org/1212.1579.

- [20] S. Parsell and J. P. Sorenson. Fast bounds on the distribution of smooth numbers. In F. Hess, S. Pauli, and M. Pohst, editors, *Proceedings of the 7th International Symposium on Algorithmic Number Theory (ANTS-VII)*, pages 168–181, Berlin, Germany, July 2006. Springer. LNCS 4076, ISBN 3-540-36075-1.
- [21] V. Ramaswami. The number of positive integers $\leq x$ and free of prime divisors x^c , and a problem of S. S. Pillai. *Duke Mathematics Journal*, 16(1):99–109, 1949.
- [22] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [23] L. Schoenfeld. Sharper bounds for the Chebyshev functions $\theta(x)$ and $\psi(x)$. II. *Mathematics of Computation*, 30(134):337–360, 1976.
- [24] J. P. Sorenson. A fast algorithm for approximately counting smooth numbers. In W. Bosma, editor, *Proceedings of the Fourth International Algorithmic Number Theory Symposium (ANTS IV)*, pages 539–549, Leiden, The Netherlands, 2000. LNCS 1838.
- [25] J. P. Sorenson. The pseudosquares prime sieve. In F. Hess, S. Pauli, and M. Pohst, editors, *Proceedings of the 7th International Symposium on Algorithmic Number Theory (ANTS-VII)*, pages 193–207, Berlin, Germany, July 2006. Springer. LNCS 4076, ISBN 3-540-36075-1.
- [26] K. Suzuki. An estimate for the number of integers without large prime factors. *Mathematics of Computation*, 73:1013–1022, 2004. MR 2031422 (2005a:11142).
- [27] K. Suzuki. Approximating the number of integers without large prime factors. *Mathematics of Computation*, 75:1015–1024, 2006.
- [28] G. Tenenbaum. *Introduction to Analytic and Probabilistic Number Theory*, volume 46 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, english edition, 1995.
- [29] C. Zhang. *An extension of the Dickman function and its application*. PhD thesis, Purdue University, 2002.

COMPUTER SCIENCES DEPARTMENT, UNIVERSITY OF WISCONSIN-MADISON

E-mail address: `bach@cs.wisc.edu`

COMPUTER SCIENCE AND SOFTWARE ENGINEERING DEPARTMENT, BUTLER UNIVERSITY

E-mail address: `sorenson@butler.edu`