Divide and Conquer

Thomas Nicely didn't set out to discover a bug in the brand new Pentium processor. It just worked out that way.

In a sense, *all* computers do division wrong.

Nicely, a professor of mathematics at Lynchburg College in Lynchburg, Virginia, is the man responsible for calling public attention to a design problem in the Pentium chip. The result was a public relations nightmare for Intel, the Pentium's manufacturer.

Introduced in 1993, the Pentium chip was touted as bringing a new level of speed and power to personal computing. By fall, 1994, Intel had sold nearly a million Pentiums, powering IBMs, Packard Bells, and other computers. At around $1000 a pop, the Pentium promised Intel a tidy profit. Then Nicely dropped his bombshell: The Pentium chip had trouble with arithmetic. When asked to divide one number by another, it sometimes gave the wrong answer.

In a sense, *all* computers do division wrong. When a computer's floating point unit, which handles numbers in decimal form, divides one number by another, it almost always makes a small error. The same thing happens with multiplication, and even with addition and subtraction. That's because computer arithmetic is usually done not with exact numbers, but with approximations having a fixed number of decimal places; every operation is rounded to give a result with the same fixed number of decimal places. For example, $1/3 = 0.33333...$ is stored not as an infinite sequence of 3's, but as some finite string.

These round-off errors are predictable, and computer scientists have developed rules for when to round up and when to round down. Floating point units typically give results that are valid to between 10 and 20 decimal places; many offer "extended precision" capabilities that guarantee more digits accuracy. Round-off errors seldom matter in routine calculations, but in scientific computations, which may involve billions of operations, the accumulation of error is a serious concern. If, for example, ten billion numbers, each "good" to ten decimal places, are added together, the computer's answer may be meaningless in every digit. In practice, there is usually about as much rounding up as down, and a statistical principle known as the random-walk phenomenon says that the

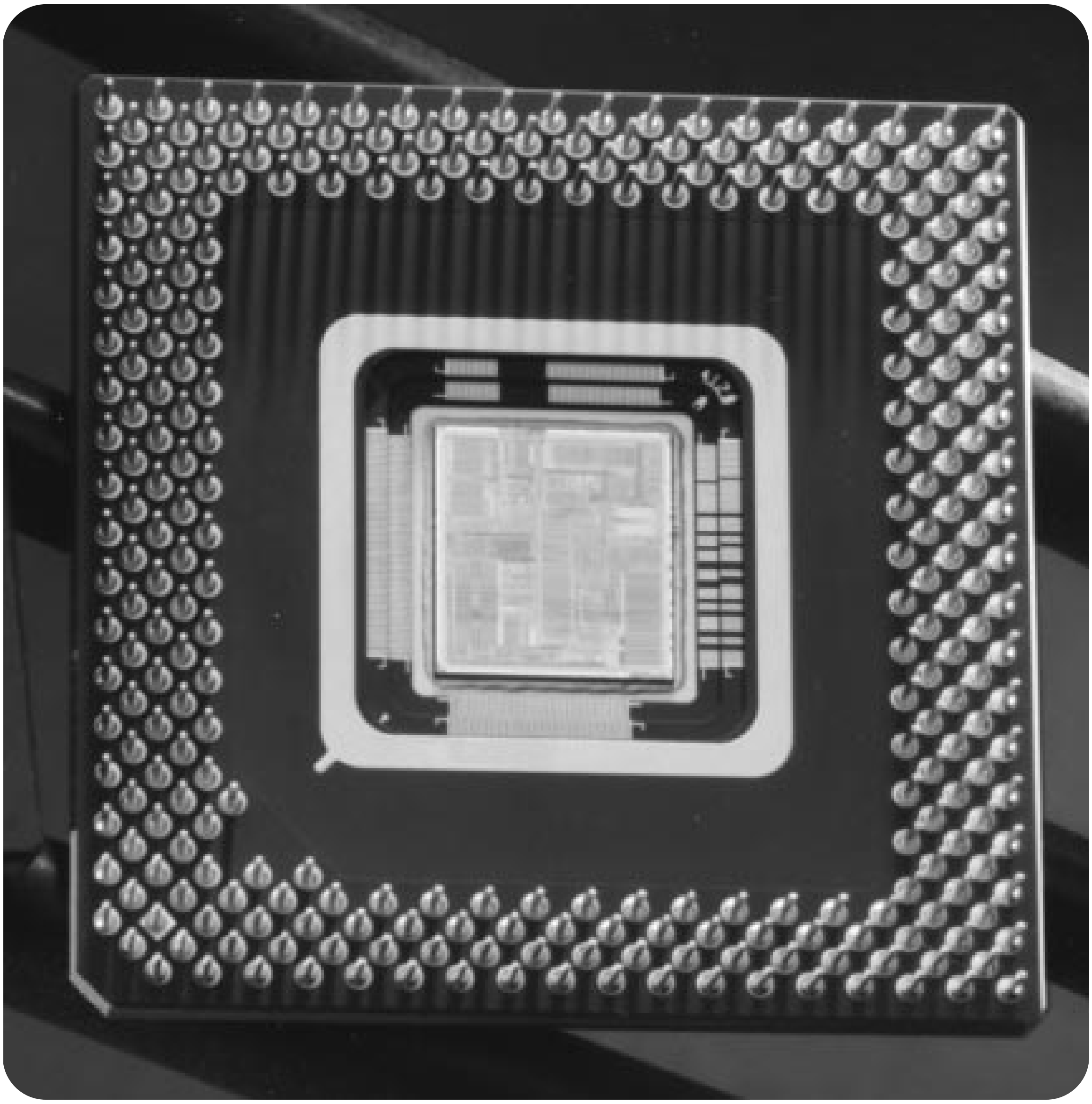


Figure 1. *The Intel Pentium® processor. (Photo courtesy of Intel Corporation.)*

result is probably good to five decimal places—which may or may not be satisfactory.

The Pentium's problem was that while it guaranteed 19 digits accuracy for each floating point operation, it sometimes delivered far fewer. The chip, as first manufactured, was missing part of its division algorithm. The absent component belonged to a "look-up" table that the chip uses for division—something like not knowing how many 6's are in 43. The Pentium's particular blind spot for division rarely displays itself; that's why the error slipped past Intel's quality control tests. When it occurs, however, the error can appear in the fifth decimal place—a performance a hundred billion times worse than the chip purports to deliver. Intel discovered the error for itself in the summer of 1994, and moved quickly to correct it. The company decided, however, not to bother owners of the defective chip with a technical advisory, figuring the error arose so seldom that nobody would be affected.

That was Intel's second mistake.

Hardware bugs are nothing new. Every machine since the abacus does quirky things now and then, as anyone whose PC just crashed can attest. But the Pentium bug was especially embarrassing, because it involved an operation most people master in grade school. Still, an error that crops up once in a billion times—who would ever notice? Or so the thinking went.

Enter Nicely, and a problem known as the Twin Prime Conjecture. Prime numbers have long fascinated mathematicians. In Book IX of the *Elements*, Euclid proved that there are infinitely many primes. His argument is simple, elegant, and compelling: Given any *finite* list of primes, their product plus 1 is a number not divisible by any of the primes in that list. But every number has at least one prime divisor, so no finite list can possibly include every prime. In 1896, Charles-Jean de la Vallée Poussin and Jacques Hadamard (independently) proved the Prime Number Theorem, which states that there are approximately $x/\ln x$ primes less

Thomas Nicely. *(Photo courtesy of Lynchburg College, Lynchburg, Virginia.)*

than any given (real) number x, where $\ln x$ is the natural logarithm of x. Thus, for example, there are approximately $100/\ln 100 \approx 22$ primes less than 100, which is not too far from the exact value of 25.

Armed with the prime number theorem, mathematicians have solved many mysteries about prime numbers, but many others remain. One especially vexing problem concerns “twin primes”—pairs of consecutive odd primes, such as 3 and 5, 5 and 7, 11 and 13, etc. The question is this: Are there infinitely many such pairs, or does the property die out somewhere down the (infinite) list of primes?

Heuristic reasoning has led number theorists to conjecture that an analog of the prime number theorem holds for twin primes: The number of such pairs up to x should be roughly proportional to $x/(\ln x)^2$. In 1919, the Norwegian mathematician Viggo Brun proved that, for large values of x, the number of twin primes is less than $100x/(\ln x)^2$. (That upper bound has more recently been brought down to around $6x/(\ln x)^2$.) But no *lower* bound has been found that would prove the infinitude of twin primes.

Brun also proved a curious theoretical result: If you reciprocate all the twin primes and add them together, forming the sum

$$(\frac{1}{3}+\frac{1}{5})+(\frac{1}{5}+\frac{1}{7})+(\frac{1}{11}+\frac{1}{13})+(\frac{1}{17}+\frac{1}{19})+\cdots,$$

the sum is *finite*—the infinite series converges to some number B, which number theorists call Brun’s sum.

Estimating Brun’s sum numerically, however, is a pain, because twin primes occur so unpredictably. Any finite portion of Brun’s sum gives an underestimate, just as any finite portion of the sum $1+\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^4}+\cdots$ gives a value a bit smaller than 2. With Brun’s sum, there’s no way of knowing how close your estimate comes. You may have stopped just short of a long prime-rich stretch of numbers, which would make your estimate way too small.

Thinking heuristically, number theorists conjectured that adding a “correction” term proportional to $1/\ln p$ onto the part of Brun’s sum that stops at $\frac{1}{p}+\frac{1}{p+2}$ will give an estimate with error less than some multiple of $1/(\sqrt{p}\ln p)$. In 1974, mathematicians Daniel Shanks and John Wrench, Jr., in the computation and mathematics

The Pentium’s problem was that while it guaranteed 19 digits accuracy for each floating point operation, it sometimes delivered far fewer.

department of the Naval Ship Research and Development Center in Bethesda, Maryland, tested this conjecture by looking at all the twin primes among the first two million prime numbers (see Figure 2). The next year, Richard Brent at the Australian National University tabulated all twin primes up to a hundred billion—there are 224,376,048 pairs—and obtained an estimate of 1.90216054 for Brun's sum (see Figure 3, page 44).

After that, most number theorists stopped worrying much about

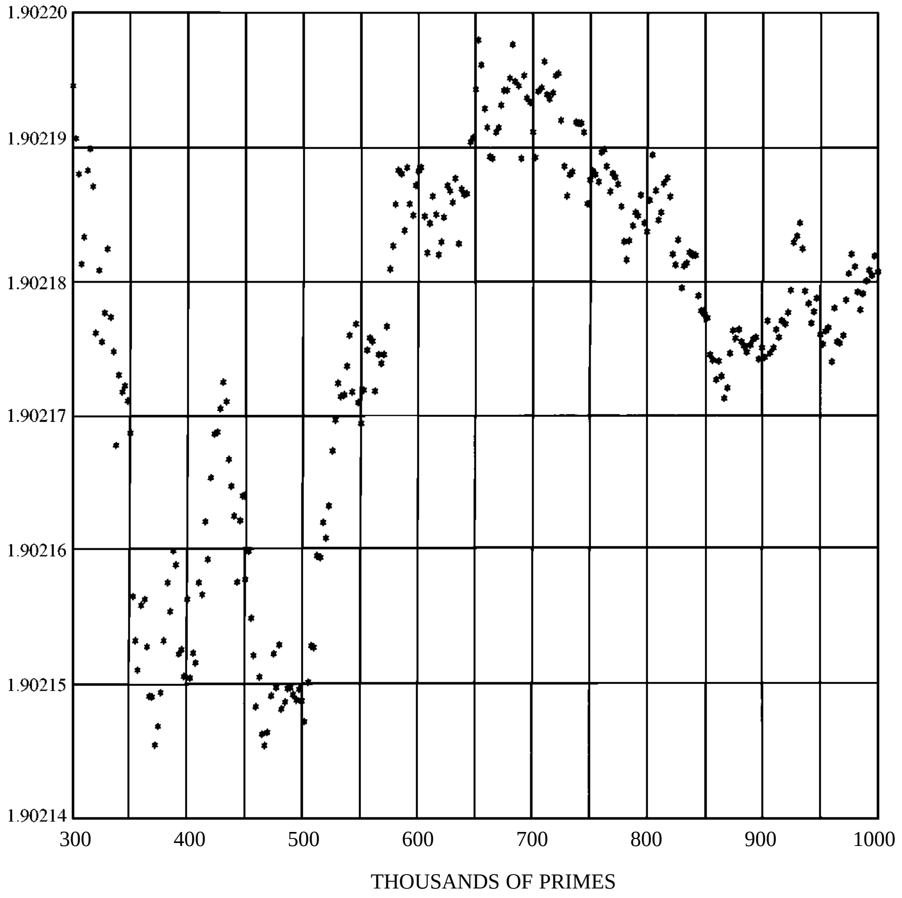
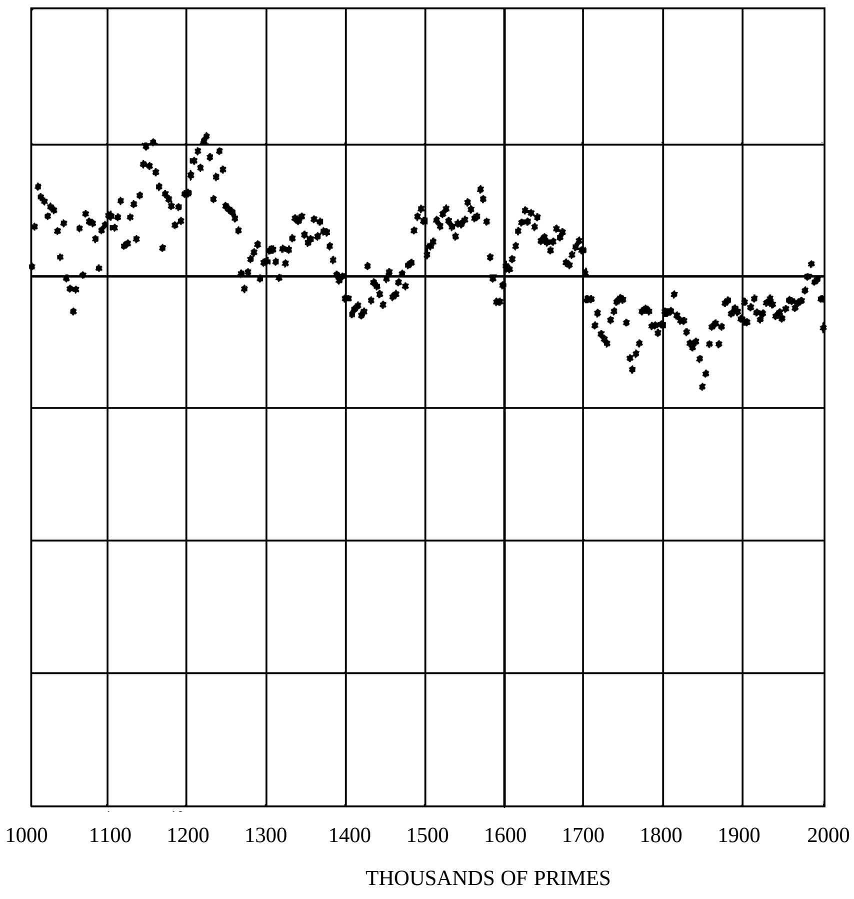


Figure 2. *Estimates of Brun's sum based on the occurrence of twin primes among the first million prime numbers (left) and the second million prime*

Brun's sum—they had plenty of other interesting problems to keep their computers busy. One was the unexpected development of cryptographic systems based on the apparent difficulty of factoring large numbers (see "The Secret Life of Large Numbers," pages

90–99). But in 1993, Thomas Nicely decided to have another go at Brun's sum.

Nicely had been looking for a problem that could be attacked with desktop computers. "I felt if I could come up with such a problem, it might serve as a model for other people working in small college environments, who don't have access to supercomputers," he says. Nicely eventually settled on studying the twin primes. He hoped to push Brent's computation up into the trillions.



numbers (right). (From "Brun's constant," Daniel Shanks and John W. Wrench, Jr., Mathematics of Computation **28***, no. 125 (1974), page 295.)*

Conceptually, there's no difficulty. The first step is to list all primes into the trillions. Nowadays this doesn't take very long. (storing the results is problematic, however—Nicely processes numbers in batches of a billion). The second step is to pick out all twin primes. Finally, one reciprocates all the survivors and adds them up, tacking on the conjectured correction term at the end. What appealed to Nicely was that the computation could be split up among several machines, each searching for primes in separate stretches.

Nicely started in 1993 with five computers, all running then-state-of-the-art "486" chips. In March, 1994, he added a Pentium machine.

The hardest problem, Nicely knew, would be deciding whether the computations were trustworthy. Accumulating round-off error was one problem: Expecting upwards of 100 billion pairs of twin primes, Nicely knew he would need to compute reciprocals to a very high accuracy to improve on Brent's estimate. He also had to worry that some error in his program—or in the operating system of the computers he planned to use—might make nonsense of the computation.

To be sure he didn't inadvertently skip any primes, Nicely checked his tallies of primes against previously published results. To be doubly sure, he decided to compute Brun's sum in two different ways: first using each computer's floating point unit to calculate each reciprocal to 19 digits, and then using an ultra-precision algorithm that had been developed by Arjen Lenstra at Bell Communications Research (Bellcore) in Morristown, New Jersey. Nicely set the ultra-precision code to work with 26 digits accuracy.

By June, he had results—and an obvious error. "The computed check value for $\pi(x)$ [the number of primes up to x] disagreed with the published value," Nicely recalls. He also observed that the round-off error in his ultra-precision calculations was accumulating much faster than expected. After a long search for mistakes in his program, he finally found a subtle error in the Borland C++ 4.02 compiler that translated things into instructions for the computer.

n	$\pi_2(n)$	$B^*(n)$
10^3	35	1.90030531
10^4	205	1.90359819
10^5	1224	1.90216329
10^6	8169	1.90191335
10^7	58980	1.90218826
10^8	440312	1.90216794
10^9	3424506	1.90216024
2×10^9	6388041	1.90215957
3×10^9	9210144	1.90215977
4×10^9	11944438	1.90215950
5×10^9	14618166	1.90215984
6×10^9	17244409	1.90216027
7×10^9	19830161	1.90216007
8×10^9	22384176	1.90216011
9×10^9	24911210	1.90216037
10^{10}	27412679	1.90216036
2×10^{10}	51509099	1.90216076
3×10^{10}	74555618	1.90216064
4×10^{10}	96956707	1.90216031
5×10^{10}	118903682	1.90216031
6×10^{10}	140494397	1.90216033
7×10^{10}	161795029	1.90216032
8×10^{10}	182855913	1.90216040
9×10^{10}	203710414	1.90216053
10^{11}	224376048	1.90216054

Figure 3. *Twin primes and Brun's sum up to 100 billion, studied by Richard Brent. (From "Irregularities in the distribution of primes and twin primes," Richard P. Brent,* Mathematics of Computation **29**, *no. 129 (1975), page 45.)*

"For some time I believed this to be the source of my woes," he says. To reassure himself, he extended the ultra-precision algorithm to 53 digits accuracy. He also began running every computation on at least two separate machines, so that he could compare results.

It was a good thing he did. "Once I worked around the compiler bug, the excess rounding error went away in the ultra-precision routine," Nicely recalls. "At that point I expected the computation to behave correctly."

It didn't.

In early October, Nicely got comparison results off a 486 machine for the twin primes up to a trillion—a computation the faster Pentium chip had already completed. "Five minutes later, I knew the error was still there, and it was worse than I had thought," he says. The ultra-precision results from the two computers differed in their last 20 decimal places. Round-off couldn't be the culprit; one of the two machines—possibly both—was doing something wrong.

Fortunately, Nicely had stored intermediate results from each machine at the end of each billion numbers, so he could quickly find a range where the two machines differed. Within a few days he had isolated the problem: For some reason, Nicely's Pentium machine had miscalculated the reciprocals of the twin primes 824,633,702,441 and 824,633,702,443. For those two numbers, the Pentium, whose floating point unit promised 19 digits accuracy, was giving only nine.

Using special-purpose debugging software, Nicely traced the error to a single instruction at the Pentium computer's assembly level; this led him to believe the culprit had to be the floating point unit itself. However, "even at that point I was trying to think of something else that could cause [the error]," such as a virus or a problem with his machine's data bus, Nicely recalls. "But eventually I was able to try the calculation on other Pentium systems, and all of them made the same error, and the only thing they had in common was the chip."

Intel's Pentium chip couldn't divide.

"It was just stunning," Nicely recalls. "I couldn't believe that the chip would have been released with that error in it." But it had been. Moreover, Nicely got no response when he contacted Intel with his discovery. Finally, he posted the news on the Internet. The Net result was a public embarrassment for the chip's manufacturer.

For some reason, Nicely's Pentium had miscalculated the reciprocals of the twin primes 824,633,702,441 and 824,633,702,443.

Intel finally acknowledged the error, but insisted that few of its customers would ever be affected. To get a replacement chip, the company first said, customers would have to demonstrate a need for one. The outcry was deafening, and Intel finally backed off, agreeing to replace the chip on demand.

Although its Pentium chip was the butt of many jokes, Intel may have the last laugh yet. The chip "sold like hotcakes" for Christmas 1994, says company spokesman Thomas Waldrop. By the middle of 1995, Intel had sold tens of millions, more than enough to offset the $475 million write-off it took for offering replacements. The chip has also seen several speed upgrades; a 150 megaHertz model is expected by late 1995. Intel also has a new chip, the P6, waiting in the wings.

Nicely could put such a chip to use. His original goal has now been met. He reports finding 135,780,321,665 pairs of twin primes in the first 100 trillion numbers, and an estimate for Brun's sum of 1.9021605778 (see Figure 4). As a result of the publicity, he wound up in touch with Martin Kutrib and Jörge Richstein at the University of Giessen in Germany, who had also been pursuing Brent's sum into the trillions. They have been trading results, which, Nicely notes, provides yet another check on the computation.

Despite all evidence to the contrary, too many people believe that if an answer comes out of a machine, it has to be right.

Indeed, says Nicely, "I'm really distrustful of any result that comes from a single machine." He now uses at least two different computers for all calculations. And with good reason: "Five more times since last October I've gotten different results from two machines," he notes. Two he traced to faulty memory chips, two to bugs in a memory manager, and one to a hard-disk failure.

To Nicely, the bugs and failures are no surprise. "They're much more prevalent—and important—than most people believe or want to believe," he says. Despite all evidence to the contrary, too many people believe that if an answer comes out of a machine, it has to be right. As computers run more and more of the equipment that directly affects our lives—everything from flight control systems to X-ray machines—computer designers will need to compensate for the unavoidable imperfections of their products. "It's something that needs to be taken a lot more seriously," says the man who took a math problem seriously enough to track down a once-in-a-billion error.

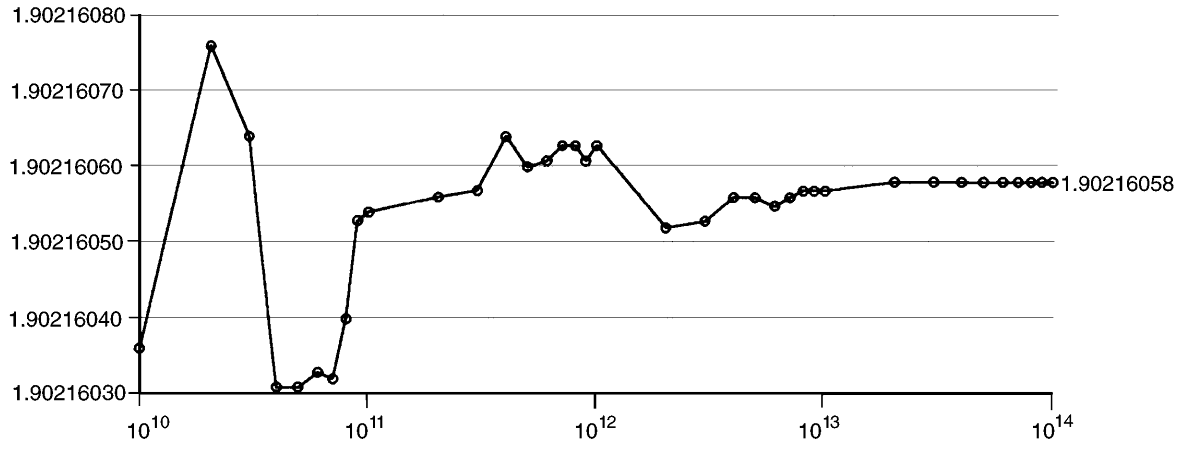


Figure 4. *Graphical form of Nicely's estimate for Brun's sum in the range from 10 billion to 100 trillion. The data points up to 100 billion were computed by Richard Brent in 1975.*