

Spigot Algorithms

Ian Stewart

26/09/2008

**Mathematics Institute
University of Warwick
Coventry CV4 7AL
UK**

"But there can't be anything new to find out about π ," said Malcolm.

"Really?" I said. Malcolm had dropped by to borrow a bag of sugar just as I was getting ready for the monthly meeting of HS π W, the Honorable Society of π -Watchers. I had mistakenly attempted to enlighten him.

"Anything interesting, I mean. After all, it must be known to billions of digits by now. What use are a few extra digits?"

"Oh dear," I said. "Is that the most imaginative property of π you can think of? A few extra digits? You think mathematical research is just a matter of calculating things a bit more accurately?"

He went all defensive. "No, of course not. But, well, I mean — π . It must be the most raked-over corner of mathematics in existence."

I grinned. "That, Malcolm my friend, is because it is one of the richest corners of mathematics in existence. Why, only six years ago the Borweins found amazing new connections between π and Ramanujan's theory of modular equations, leading to new superconvergent approximations to π ."

"Superconvergent?"

"Giving five times more decimal digits at every step."

"And what did people *do* with this amazing new discovery?"

"Calculated a few more decimal digits of π ," I admitted. "Well, a few billion more. But that was just a test. And it's not easy, you know, to compute numbers to billions of decimal places."

"Why not?"

"Well, the standard computer languages generally work with ten decimal places, or twenty in double precision. You have to use totally different software to handle very long sequences of digits." He nodded. "Which is why tonight's topic is so exciting."

He picked up the bag of sugar. "What topic?"

"Spigot algorithms."

He stared at me. "You might just as well say 'cheese boondoggles', for all that means anything to me."

"I'm going to show you how to calculate π , one digit at a time, using only whole number arithmetic," I said. I grabbed a piece of kitchen paper and a pen, and wrote on it:

A		1	2	3	4	5	6	7	8	9	10	11	12
B		3	5	7	9	11	13	15	17	19	21	23	25
		2	2	2	2	2	2	2	2	2	2	2	2

"Those are your starting numbers. Now, just follow my instructions. Multiply everything in the row below the horizontal line by 10."

"Like this?" He wrote

20 20 20 20 20 20 20 20 20 20 20 20 20

"Doesn't look much like _ to me."

"Not yet. Now, make three blank rows. The top will hold the 'carry', the middle one the 'sum', and the bottom one the 'remainder'. You start at the right hand end, and that carry is 0, so it looks like this:

A		1	2	3	4	5	6	7	8	9	10	11	12
B		3	5	7	9	11	13	15	17	19	21	23	25
		2	2	2	2	2	2	2	2	2	2	2	2
		20	20	20	20	20	20	20	20	20	20	20	20
carry													0

Now, add the carry to the number above, to get the sum 20. So far so good, but now comes the tricky part. Look up to the top of the column, in row B. What do you see?"

"Um — 25."

"Right. Now, you find the remainder r and quotient q on dividing 20 by 25. Those are?"

"Er, 25 into 20 goes — zero times, with remainder 20, I guess."

"Good. So $r = 20$, $q = 0$. Write in the remainder row in the last column. Take q and multiply it by the number in row A at the top of the last column, which is?"

"12."

"OK. Now $q = 0$ here, and $12 \cdot 0 = 0$. That's the *carry* digit for the next column to the left." Like this:

A		1	2	3	4	5	6	7	8	9	10	11	12
B		3	5	7	9	11	13	15	17	19	21	23	25
		2	2	2	2	2	2	2	2	2	2	2	2
		20	20	20	20	20	20	20	20	20	20	20	20
carry												0	0
sum													20
remainder													20

"Now, I want you to repeat the same operations on the next column, the one you've just put the carry 0 in. Only this time you want the remainder on dividing by 23, the number in row B that corresponds to the new column; and the quotient gets multiplied by 11, the number in row A. And carried one space left, again." Like this:

A		1	2	3	4	5	6	7	8	9	10	11	12
B		3	5	7	9	11	13	15	17	19	21	23	25
		2	2	2	2	2	2	2	2	2	2	2	2
		20	20	20	20	20	20	20	20	20	20	20	20
carry												0	0
sum												20	20
remainder												20	20

"And again, but now you take the remainder on division by 21 and multiply the quotient by 10." Like this:

A		1	2	3	4	5	6	7	8	9	10	11	12
B		3	5	7	9	11	13	15	17	19	21	23	25
		2	2	2	2	2	2	2	2	2	2	2	2
		20	20	20	20	20	20	20	20	20	20	20	20
carry												0	0
sum												20	20
remainder												20	20

"It's not very interesting," said Malcolm.

"Not yet. Give it a while. Same trick again on the new column, but now take the remainder on dividing by 19 and multiply the quotient by 9. Now it starts to change." This time the sum, 20, leaves remainder 1 on division by 19. The quotient is 1, and multiplied by 9 this gives 9. Like this:

		1	2	3	4	5	6	7	8	9	10	11	12
A													
B		3	5	7	9	11	13	15	17	19	21	23	25
<hr/>													
		2	2	2	2	2	2	2	2	2	2	2	2
		20	20	20	20	20	20	20	20	20	20	20	20
carry								9	0	0	0	0	
sum									20	20	20	20	
remainder									1	20	20	20	

"OK, you should have the idea by now," I told him. Repeat column by column. Always divide by the entry in row B of that column to find a remainder and a quotient. Multiply the quotient by row A and carry left." For instance, the next few steps are to add 20 to 9 to get 29. Divide by 17 (row B) to get remainder 12 and quotient 1. Then the quotient times 8 (row A) gives a carry of $1 \times 8 = 8$, and the table looks like this:

		1	2	3	4	5	6	7	8	9	10	11	12
A													
B		3	5	7	9	11	13	15	17	19	21	23	25
<hr/>													
		2	2	2	2	2	2	2	2	2	2	2	2
		20	20	20	20	20	20	20	20	20	20	20	20
carry								8	9	0	0	0	0
sum									29	20	20	20	20
rem									12	1	20	20	20

Continuing in this manner, we get:

		1	2	3	4	5	6	7	8	9	10	11	12
A													
B		3	5	7	9	11	13	15	17	19	21	23	25
<hr/>													
		2	2	2	2	2	2	2	2	2	2	2	2
		20	20	20	20	20	20	20	20	20	20	20	20
carry		10	12	12	12	10	12	7	8	9	0	0	0
sum		30	32	32	32	30	32	27	28	29	20	20	20
rem			2	2	4	3	10	1	13	12	1	20	20

"What do I do now?" asked Malcolm. "There's nothing to divide by."

"Take the sum 30 (marked in bold), and think of it as a remainder of 0 and a carry of 3 (underlined), just as you would in decimal notation. Write down the 0 in the remainder column, and write the 3 on a separate sheet of paper."

"OK. This is silly."

"No it isn't. The spigot has just dispensed the first digit of $_$, which is 3. You've written it on your paper, right?"

"You mean —"

"Yes. Carry out the same manoeuvres again, but starting with the new list of remainders instead of all those 2's, and you'll get the next digit." It went like this:

		1	2	3	4	5	6	7	8	9	10	11	12	
A														
B		3	5	7	9	11	13	15	17	19	21	23	25	
		2	2	2	2	2	2	2	2	2	2	2	2	
rem		0	2	2	4	3	10	1	13	12	1	20	20	20
$_10$		0	20	20	40	30	100	10	130	120	10	200	200	200
carry		13	20	33	40	65	48	98	88	72	150	132	96	0
sum		<u>13</u>	40	53	80	95	148	108	218	192	160	332	296	200
rem		3	1	3	3	5	5	4	8	5	8	17	20	0

The final sum, 13 (bold), becomes a remainder of 3 and a carry of 1. That 1, underlined, is the next digit of $_$ to pop out of the spigot. So far, the algorithm has given us $_ = 3.1$.

Just to check you've got the idea, here's the next stage:

		1	2	3	4	5	6	7	8	9	10	11	12	
A														
B		3	5	7	9	11	13	15	17	19	21	23	25	
		2	2	2	2	2	2	2	2	2	2	2	2	
rem		3	1	3	3	5	5	4	8	5	8	17	20	0
$_10$		30	10	30	30	50	50	40	80	50	80	170	200	0
carry		11	24	30	40	40	42	63	64	90	120	88	0	0
sum		<u>41</u>	34	60	70	90	92	103	144	140	200	258	200	0
rem		1	1	0	0	0	4	12	9	4	10	6	16	0

Now the next digit, 4, pops out. The spigot has delivered $_ = 3.14$. If you try again, you should get 3.141.

"How many digits does it produce?" asked Malcolm.

"Just the first four," I said. "But if you want n digits, you just start with $3n+1$ consecutive 2's. You have more columns, and you extend rows A and B in the obvious

way; but the calculations are simple operations on small integers. It *never* gets very complicated."

"Is that how people used to calculate the digits of π ?" asked Malcolm. "I remember some chap called Ludolph van Ceulen calculated π to 20 decimal places."

"Yes, in 1596. The German's have called π 'Ludolph's number' ever since. No, that's not how he did it. In fact, with the spigot algorithm you could compute 20 digits of π by hand in just a few hours, using a mere 61 columns." I paused. "Actually there's one extra twist to the method which I haven't told you yet, but it doesn't show up until the 32nd decimal place."

Malcolm shook his head in wonderment. "How does it work?"

"That's an interesting story," I said. "The irony of it all is that the method wasn't discovered until 1968, by A.H.J.Sale, only he did it for the number e , the base of natural logarithms. The extension to π was published in 1991 by Stanley Rabinowitz of Westford, Massachusetts. Their investigations were motivated by the availability of modern computers — but by then computers had extended Ludolph's calculation far beyond anything he would have dared to imagine."

"OK then," said Malcolm.

"OK what?"

"Let's have the interesting story."

"Oh. Right. Well, it starts with standard decimal notation, of course. That's where the 'multiply by 10' step comes from."

"Naturally."

"You'll see. In decimal notation, you can write a number like $e = 2.718281\dots$ as a nested algebraic formula, like this:

$$2.718281 = \mathbf{2} + \frac{1}{10} (\mathbf{7} + \frac{1}{10} (\mathbf{1} + \frac{1}{10} (\mathbf{8} + \frac{1}{10} (\mathbf{2} + \frac{1}{10} (\mathbf{8} + \frac{1}{10} (\mathbf{1} + \dots) \dots)))$$

You can see the successive digits, in boldface; and the fractions $\frac{1}{10}$ used at each stage

represent the arithmetic *base*, which we consider to be the sequence $\mathbf{a} = (\frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \dots)$.

I admit it's not quite the usual way to think of decimals, but it's convenient here.

"Now, you can use other bases instead. In particular you can use so-called *mixed* bases, where the terms of the base sequence are different. A useful one is base

$$= (\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \dots).$$

In this base a number $a_0.a_1a_2a_3a_4\dots$ is interpreted as the sum

$$a_0 + \frac{1}{2}(a_1 + \frac{1}{3}(a_2 + \frac{1}{4}(a_3 + \frac{1}{5}(a_4 + \dots))))$$

or equivalently as

$$a_0 + \frac{1}{2}a_1 + \frac{1}{2 \cdot 3}a_2 + \frac{1}{2 \cdot 3 \cdot 4}a_3 + \frac{1}{2 \cdot 3 \cdot 4 \cdot 5}a_4 + \dots$$

"Great, but what use is it?"

"Well, in mixed base notation some irrational numbers have very simple expressions. For example in base b the number e can be written as

$$1.111111\dots$$

because of the well known series

$$e = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \frac{1}{1 \cdot 2 \cdot 3 \cdot 4} + \frac{1}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5} + \dots$$

"I'm beginning to see the light," said Malcolm. "It's just a question of converting

from base b to base 10 — well, base $(\frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \dots)$. The answer is trivial in base b , and the spigot algorithm is just the conversion step."

"Precisely. The main problem you have to worry about is the uniqueness of the representation $a_0.a_1a_2a_3a_4\dots$ in a mixed base. If the same number is represented in more than one way, you have to take care. In a constant base it's easy: for instance in base $(\frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \dots)$ each digit a_j must be between 0 and 9. Even then you get a slight lack of uniqueness, because $0.999999\dots = 1.000000\dots$."

"I thought 0.999999... was a bit less than 1."

"No. It's less if you *stop* the sequence anywhere, but it's the same if the sequence goes on forever."

"Oh."

"In base b the condition for a unique representation is that $0 \leq a_m < b$ for each m . Now, if you think about how to convert between different bases, you come up with Sale's spigot algorithm for e , which runs as follows if you want n decimal places:

- (1) Let the first digit be 2 (equal to $1 + \frac{1}{1}$ to avoid having the improper fraction $\frac{1}{1}$ in the base) followed by an array of $n+1$ 1's.
- (2) Repeat $n+1$ times:
 - (2.1) Multiply each entry by 10.

- (2.2) Starting from the right, divide the m th entry by $m+1$.
- (2.3) Carry the quotient one place left.
- (2.4) The final carry, from column 1, is the next digit of e .

Here's the start of the calculation when $n = 7$. I've written it with rows marked A and B to show the similarity to the algorithm for e . Step (2.2) is the 'divide by row B' step, and step (2.3) is the 'multiply quotient by row A' step. But here row A is all 1's, so there's no need to multiply.

A		1	1	1	1	1	1	1
B		2	3	4	5	6	7	8
<hr/>								
start	2	1	1	1	1	1	1	1
_10		10	10	10	10	10	10	10
carry	7	4	3	2	1	1	1	0
sum		14	13	12	11	11	11	10
rem.		0	1	0	1	5	4	2
_10		0	10	0	10	50	40	20
carry	1	3	0	3	9	6	2	0
sum		3	10	3	19	56	42	20
rem.		1	1	3	4	2	0	4
_10		10	10	30	40	20	0	40
carry	8	6	9	8	3	0	5	0
sum		16	19	38	43	20	5	40
rem.		0	1	2	3	2	5	0
_10		0	10	20	30	20	50	0
carry	2	5	6	6	4	7	0	0
sum		5	16	26	34	27	50	0
rem.		1	1	2	4	3	1	0

and so on.

One warning: because of rounding-off errors in decimal notation, it's worth making n a bit bigger than the algorithm specifies.

"OK, I pretty much follow that," said Malcolm. "In base 10, the way to get the next digit is to chop off the integer part, multiply by 10, and take the integer part of that. You're doing exactly the same in base b arithmetic, but there are some extra stages to get the numbers back into the correct form for a unique representation. That's what all those remainders and quotients are for."

"Absolutely."

"But it's not a spigot algorithm. It's a spreadsheet algorithm."

"That's true," I said. "It would be easy to implement it on a spreadsheet."

He nodded vigorously. "Can I try something else?"

"If you want to stick to base b , I recommend

$$\cosh(1) = \frac{1}{2}(e + e^{-1}) = 1.543080568313599\dots$$

The first row should look like

1 0 1 0 1 0 1 0 1 0

where the boldface 1 is the first digit of the answer and the rest is the first row of the array." To get you started, here are the first few steps:

A		1	1	1	1	1	1	1	1	1
B		2	3	4	5	6	7	8	9	10 11
<hr/>										
		1	0	1	0	1	0	1	0	1 0
x10		10	0	10	0	10	0	10	0	10 0
carry	5	0	2	0	1	0	1	0	1	0 0
sum		10	2	10	1	10	1	10	1	10 0
rem		0	2	2	1	4	1	2	1	0 0
x10		0	20	20	10	40	10	20	10	0 0
carry		8	5	3	6	1	2	1	0	0 0
sum		18	25	23	16	41	12	21	10	0 0
rem		0	1	3	1	5	5	5	1	0 0
x10		0	10	30	10	50	50	50	10	0 0

"What about $_$?" asked Malcolm.

"Hmm. Now you're getting interested. I thought there was nothing new to learn about $_$."

"I admit it, I was wrong. Go on."

"OK. The first problem is to find a series for $_$ that lends itself to the spigot treatment. It doesn't seem possible to work to base b any more — you don't get any nice pattern in the base- b digits of $_$. But it can be shown that in base

$$c = \left(\frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \frac{4}{9}, \dots \right)$$

we have $_ = 2.222222\dots$. In other words, the series expansion

$$_ = 2 + \frac{1}{3} \left(2 + \frac{2}{5} \left(2 + \frac{3}{7} \left(2 + \frac{4}{9} \left(2 + \dots \right) \dots \right) \dots \right) \dots \right)$$

is valid."

"I'll take your word for it."

"Fine. You'll notice that the terms in base c are the fractions formed from rows A and B of the table used in the spigot algorithm for $_$. Unfortunately, there's one snag with base c . The natural restriction on digits is that the m th digit satisfies $0 \leq a_m \leq 2m$.

But then the representation of numbers is not quite unique. The simplest way round this is to temporarily hold some tentative correction digits, called *predigits*, and adjust them when necessary."

Rabinowitz's spigot algorithm for π to n digits then looks like this:

(1) Let $2, 2, 2, 2, 2, \dots, 2$ be an array of 2's of length $\lceil 10n/3 \rceil$ where $\lceil \cdot \rceil$ is the integer part.

(2) Repeat n times:

(2.1) Multiply each entry by 10.

(2.2) Starting from the right, find the remainder r and quotient q on dividing the m th element of the array by 2^{m-1} . Leave r in place.

(2.3) Multiply q by $m-1$ and carry one place left.

(2.4) When you get to the leftmost entry, reduce it modulo 10. The quotient Q is the new predigit of π , the remainder staying in place.

(2.5) If Q is neither 9 nor 10, release the held predigits as true digits of π and hold Q .

(2.6) If $Q = 9$, add Q to the queue of held predigits.

(2.7) If $Q = 10$, then set the current predigit to 0 and hold it; increase all other held predigits by 1 (9 becoming 0); release as true digits of π all but the current held predigit.

For more details of how to do this, see the article by Rabinowitz and Wagon in *Further Reading*, which also lists a Pascal program. Using it, you can compute π to 1000 digits in a few minutes.

"Wow," said Malcolm. "Is the Honorable Society of π -Watchers open for new members?" He was hooked.

"Of course. But you have to bring some new item of information to be allowed to join. "

"What did *you* bring?"

"A very obscure integral that evaluates to $(\pi - 355/113)^4$."

"Oh." He thought for a while. "OK, got it."

"Already? A new fact about π ?"

"Yes. I've worked out a spigot algorithm for all the digits to the *left* of the decimal point."

"Which is?"

"Start with 3. Then multiply by 0 to get the previous digit, and repeat indefinitely."

"You're in," I said.

"For something that silly?"

"HS_W is very short of cash," I said. "We need every member we can get."

FURTHER READING

W.W.Rouse Ball, *Mathematical Recreations and Essays*, MacMillan, New York 1959.

J.M.Borwein, P.B.Borwein, and D.H.Bailey, Ramanujan, modular equations, and approximations to pi, or How to compute one billion digits of pi, *American Mathematical Monthly* **96** (1989) 201-219.

Stanley Rabinowitz, Abstract 863-11-482, a spigot algorithm for π , *Abstracts of the American Mathematical Society* **12** (1991) 30.

Stanley Rabinowitz and Stan Wagon, a spigot algorithm for the digits of π , *American Mathematical Monthly* **102** (1995) 195-203.

A.H.J.Sale, The calculation of e to many significant digits, *Computing Journal* **11** (1968) 229-230.