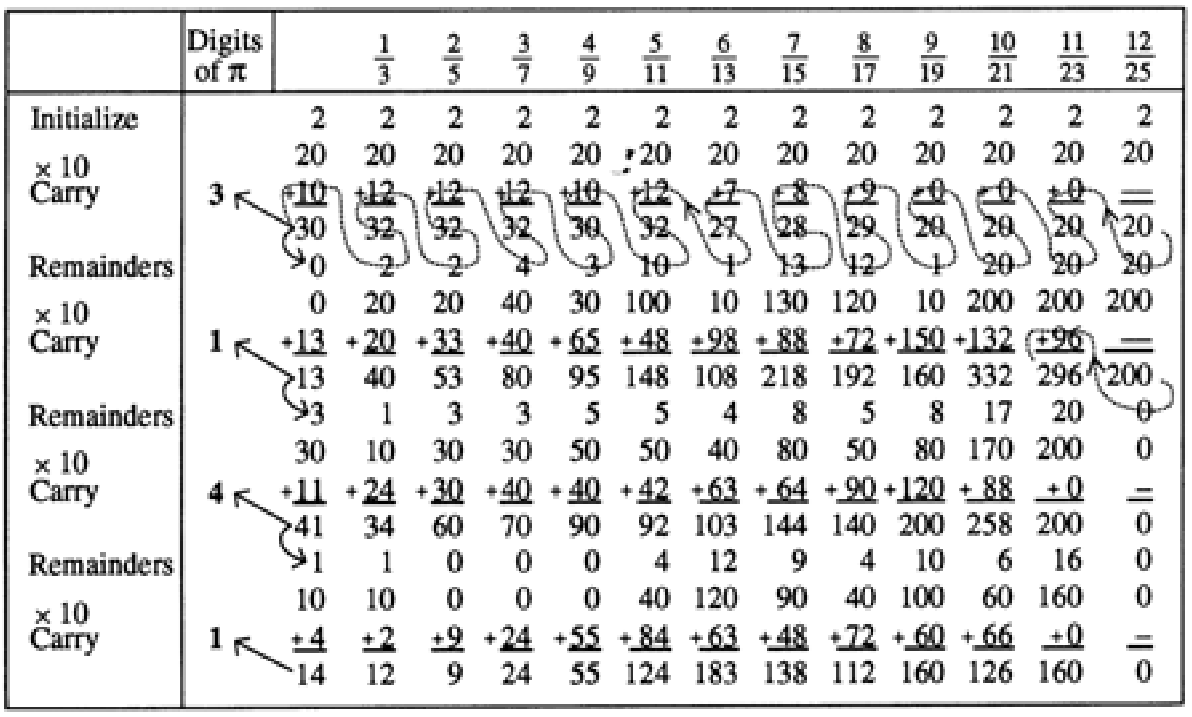
A Spigot Algorithm for the Digits of Pi

Stanley Rabinowitz and Stan Wagon

It is remarkable that the algorithm illustrated in Table 1, which uses no floating-point arithmetic, produces the digits of π. The algorithm starts with some 2s, in columns headed by the fractions shown. Each entry is multiplied by 10. Then, starting from the right, the entries are reduced modulo *den*, where the head of the column is *num*/*den*, producing a quotient q and remainder r. The remainder is left in place and $q \times num$ is carried one column left. This reduce-and-carry is continued all the way left. The tens digit of the leftmost result is the next digit of π. The process continues with the multiplication of the remainders by 10, the reductions modulo the denominators, and the augmented carrying.

TABLE 1. The workings of an algorithm that produces digits of π. The dashed line indicates the key step: starting from the right, entries are reduced modulo the denominator of the column head $(25, 23, 21, \ldots, \text{resp.})$, with the quotients, after multiplication by the numerator $(12, 11, 10, \ldots)$, carried left. For example, the 20 in the $\frac{9}{19}$'s column yields a remainder of 1 and a left carry of $1 \cdot 9 = 9$. After the leftmost carries, the tens digits are 3, 1, 4, 1. To get more digits of π one must start with a longer string of 2s.



	Digits of π	$\frac{1}{3}$	$\frac{2}{5}$	$\frac{3}{7}$	$\frac{4}{9}$	$\frac{5}{11}$	$\frac{6}{13}$	$\frac{7}{15}$	$\frac{8}{17}$	$\frac{9}{19}$	$\frac{10}{21}$	$\frac{11}{23}$	$\frac{12}{25}$
Initialize		2	2	2	2	2	2	2	2	2	2	2	2
× 10		20	20	20	20	20	20	20	20	20	20	20	20
Carry	3	+10	+12	+12	+12	+10	+12	+7	+8	+9	+0	+0	+0	—
		30	32	32	32	30	32	27	28	29	20	20	20	20
Remainders		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	1	+13	+20	+33	+40	+65	+48	+98	+88	+72	+150	+132	+96	—
		13	40	53	80	95	148	108	218	192	160	332	296	200
Remainders		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	4	+11	+24	+30	+40	+40	+42	+63	+64	+90	+120	+88	+0	—
		41	34	60	70	90	92	103	144	140	200	258	200	0
Remainders		1	1	0	0	0	4	12	9	4	10	6	16	0
× 10		10	10	0	0	0	40	120	90	40	100	60	160	0
Carry	1	+4	+2	+9	+24	+55	+84	+63	+48	+72	+60	+66	+0	—
		14	12	9	24	55	124	183	138	112	160	126	160	0

This algorithm is a "spigot" algorithm: it pumps out digits one at a time and does not use the digits after they are computed. Moreover, the digits are generated without any use of high-precision (or low-precision) operations on floating-point real numbers; the entire algorithm uses only ordinary integer arithmetic on relatively small integers. For example, to obtain the first 5,000 digits of π requires only arithmetic operations on integers less than 600,000,000. Although high-precision floating-point routines are built up from integer operations, the algorithms in this paper are quite simple and do not simulate floating-point computations.

In order to motivate the π-algorithm, we first discuss the much simpler case of e, for which a spigot algorithm was discovered by Sale [**Sale**]. His algorithm is the basis of the discussion in §1.

1. A NUMBER SYSTEM IN WHICH e's DIGITS ARE PERIODIC.

A real number's decimal representation may be interpreted as an infinitely nested expression; for example:

$$\sqrt{2} = 1.41421356\ldots = \mathbf{1} + \frac{1}{10}\left(\mathbf{4} + \frac{1}{10}\left(\mathbf{1} + \frac{1}{10}\left(\mathbf{4} + \frac{1}{10}\left(\mathbf{2} + \frac{1}{10}\left(\mathbf{1} + \cdots\right)\right)\right)\right)\right).$$

Some interesting and useful representations may be obtained if we change the base-sequence, which in the case above is $\left(\frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \ldots\right)$. For example, using the base $\mathbf{b} = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \ldots\right)$ yields the following form, called a *mixed-radix* representation (see [**Knu**, §4.1]):

$$a_0 + \frac{1}{2}\left(a_1 + \frac{1}{3}\left(a_2 + \frac{1}{4}\left(a_3 + \frac{1}{5}\left(a_4 + \frac{1}{6}\left(a_5 + \cdots\right)\right)\right)\right)\right),$$

where the a_i (the *digits*) are nonnegative integers. If $0 \le a_i \le i$ for $i \ge 1$, the representation is called *regular*. Mixed-radix representations will be denoted by $(a_0; a_1, a_2, a_3, a_4, \ldots)_\mathbf{b}$. For base $\mathbf{b}$, every positive real number has a regular representation and representations are unique provided we exclude representations that terminate with maximal digits (otherwise, for example, $\frac{1}{2} = (0; 1, 0, 0, \ldots)_\mathbf{b} = (0; 0, 2, 3, 4, 5, 6, \ldots)_\mathbf{b}$); from now on and for all bases, we exclude such representations. The proof of the following Lemma is in Appendix 1.

Lemma 1

(a) *If* $i \ge 1$, $(0; 0, 0, \ldots, 0, a_i, a_{i+1}, \ldots)_\mathbf{b} < \frac{1}{i!}$; *in particular,* $(0; a_1, a_2, a_3, a_4, \ldots)_\mathbf{b} < 1$.

(b) *Representations using the mixed-radix base* $\mathbf{b}$ *are unique.*

(c) *The integer part of* $(a_0; a_1, a_2, a_3, a_4, \ldots)_\mathbf{b}$ *is* a_0 *and the fractional part is* $(0; a_1, a_2, a_3, a_4, \ldots)_\mathbf{b}$.

In this number system some irrationals become periodic. For example $e = (2; 1, 1, 1, 1, \ldots)_\mathbf{b}$; this is just a restatement of the infinite series $\sum \frac{1}{i!}$ as

$$1 + \frac{1}{1}\left(1 + \frac{1}{2}\left(1 + \frac{1}{3}\left(1 + \frac{1}{4}\left(1 + \frac{1}{5}\left(1 + \cdots\right)\right)\right)\right)\right).$$

Rational numbers in this system correspond to digit-sequences that terminate (Appendix 1, Lemma 2).

The decimal digits of a real number x in $[0, 10)$ can be obtained by taking the integer part of x, multiplying its fractional part by 10, taking the integer part of the result, multiply the resulting fractional part by 10, and so on. In some mixed-radix bases, this is especially simple. If $x = (a_0; a_1, a_2, \ldots, a_n)_{\mathbf{b}}$, then $10x = (10a_0; 10a_1, 10a_2, 10a_3, \ldots, 10a_n)_{\mathbf{b}}$. The latter may not be a regular expression: some digits may be too big. But we can decrease digits by reducing them modulo i, where i is the denominator of the corresponding element of $\mathbf{b}$. Starting these reductions at the right end, we carry the quotients left, eventually getting the regular representation of $10x$. Thus multiplying by 10 is algorithmically straightforward. Taking the integer and fractional parts for $\mathbf{b}$-representations is also easy, thanks to Lemma 1(c).

We can now give the algorithm to get the first n base-10 digits of e. A proof of correctness — the error analysis showing that $n + 2$ mixed-radix digits suffice[1] to get n base-10 digits — is given as Lemma 3 in Appendix 1.

```
**Algorithm** *e-spigot*
  1. *Initialize:* Let the first digit be 2 and initialize an array $A$ of length $n + 1$ to
     $(1, 1, 1, \ldots, 1)$.
  2. Repeat $n - 1$ times:
       *Multiply by 10:* Multiply each entry of $A$ by 10.
       *Take the fractional part:* Starting from the right, reduce the $i$th entry of $A$ modulo
         $i + 1$, carrying the quotient one place left.
       *Output the next digit:* The final quotient is the next digit of $e$.
```

The first few steps of this algorithm, starting with an array of 10 1s (this corresponds to 11 mixed-radix digits, good for 9 digits of e; only 5 are shown), are displayed in Table 2.

[1] Any digit-producing algorithm for a presumed-normal number x suffers from a drawback that, although unlikely, can impinge on the result. If x is between 1 and 10 and the algorithm says that the first 100 digits of x are, say, $4, 6, 5, 0, 7, \ldots, 3, 9, 9, 9, 9, 9$ then one cannot be sure that the last 6 digits are correct. They will be the digits of a certain approximation to x that is within $5 \cdot 10^{-100}$ of the true value. One cannot simply go farther until a non-9 is reached, because memory allocations must be made in advance. The user must realize that a terminating string of 9s is a red flag concerning those digits and even with no 9s, the last digit might be incorrect. In practice, one might ask for, say, 6 extra digits, reducing the odds of this problem to one in a million.

TABLE 2. The workings of a spigot algorithm for the digits of e (in bold). The reductions in the column headed $\frac{1}{i}$ are performed modulo i. The leftmost base-10 real numbers are the values of the rows viewed as mixed-radix representations. Since only 11 mixed-radix digits start the algorithm, the first base-10 number is only an approximation to e.

Base 10		1/2	1/3	1/4	1/5	1/6	1/7	1/8	1/9	1/10	1/11
2.718281826...	**2**	1	1	1	1	1	1	1	1	1	1
7.18281826...		10	10	10	10	10	10	10	10	10	10
carries	**7**	+3	+3	+2	+1	+1	+1	+1	+1	+0	--
		14	13	12	11	11	11	11	11	10	10
0.18281826...		0	1	0	1	5	4	3	2	0	10
1.8281826...		0	10	0	10	50	40	30	20	0	100
carries	**1**	+3	+0	+3	+9	+6	+4	+2	+0	+9	--
		3	10	3	19	56	44	32	20	9	100
0.8281826...		1	1	3	4	2	2	0	2	9	1
8.281826...		10	10	30	40	20	20	0	20	90	10
carries	**8**	+6	+9	+8	+3	+2	+0	+3	+9	+0	--
		16	19	38	43	22	20	3	29	90	10
0.281826...		0	1	2	3	4	6	3	2	0	10
2.81826...		0	10	20	30	40	60	30	20	0	100
carries	**2**	+5	+6	+7	+8	+9	+4	+2	+0	+9	--
		5	16	27	38	49	64	32	20	9	100
0.81826...		1	1	3	3	1	1	0	2	9	1

2. A SPIGOT FOR DIGITS OF π.

The ideas of §1 lead to a spigot algorithm for π, but there are additional complexities and additional interesting questions that distinguish π from e. Our starting point is the following moderately well-known series:

$$\pi = \sum_{i=0}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!} .$$

This series can be derived from the Wallis product for π; another approach uses an acceleration technique called Euler's transform applied to the series $\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \cdots$. These proofs, together with three others and references to earlier sources, may be found in [**Li**]. We let $k!!$ denote the product $1 \cdot 3 \cdot 5 \cdots k$ for odd integers k; then the series is equivalent to

$$\frac{\pi}{2} = \sum_{i=0}^{\infty} \frac{i!}{(2i+1)!!} = 1 + \frac{1}{3} + \frac{1 \cdot 2}{3 \cdot 5} + \frac{1 \cdot 2 \cdot 3}{3 \cdot 5 \cdot 7} + \cdots ,$$

which expands to become

$$\frac{\pi}{2} = 1 + \frac{1}{3}\left(1 + \frac{2}{5}\left(1 + \frac{3}{7}\left(1 + \frac{4}{9}\left(1 + \cdots\right)\right)\right)\right).$$

This last expression leads to the mixed-radix base $\mathbf{c} = \left(\frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \frac{4}{9}, \ldots\right)$, with respect to which π is simply $(2; 2, 2, 2, 2, 2, \ldots)_{\mathbf{c}}$. For a regular representation in base $\mathbf{c}$, the digit in the ith place must lie in the interval $[0, 2i]$. Unfortunately, base $\mathbf{c}$ is less accommodating than $\mathbf{b}$.

Lemma 4 (Proof in Appendix 1). *The base-*$\mathbf{c}$ *number with maximal digits,* $(0; 2, 4, 6, 8, \ldots)$*, represents* 2*; hence regular representations of the form* $(0; a, b, c, \ldots)_{\mathbf{c}}$ *lie between* 0 *and* 2*.*

Lemma 4 implies that $\mathbf{c}$-representations are not unique. For example, $(0; 0, 4, 6, 8, \ldots)_{\mathbf{c}}$ $= 2 - \frac{2}{3} = \frac{4}{3}$, whence $(0; 0, 2, 3, 4, \ldots)_{\mathbf{c}} = \frac{2}{3} = (0; 2, 0, 0, 0, \ldots)_{\mathbf{c}}$. More relevant algorithmically, integer and fractional parts using $\mathbf{c}$ are not straightforward, as they are for $\mathbf{b}$. The integer part of $(a_0; a_1, a_2, \ldots)_{\mathbf{c}}$ is either a_0 or $a_0 + 1$ according as $(0; a_1, a_2, \ldots)$ is in $[0, 1)$ or $[1, 2)$. This problem is surmounted by leaving the units digit of a_0 in place during the next iteration and calling the tens digit of a_0 a *predigit.* The predigits must be temporarily held because occasionally (once every 20 iterations, roughly) the next predigit is a 10; this will happen when the carry, which is between 0 and 19, is greater than 10 and, simultaneously, the leftover units digit of a_0 is 9, which becomes 90 in the multiply-by-10 step. This event requires that the held number be increased by 1 before being released. Specific details of the algorithm follow; the presentation at the beginning of this paper sidestepped the problem of the occasional 10. The proof that $\lfloor 10n/3 \rfloor$ mixed-radix digits suffice for n digits of π is in Appendix 1 (Lemma 5). Appendix 2 contains a Pascal implementation of this algorithm.

Algorithm π-*spigot*

1. *Initialize:* Let $A = (2, 2, 2, 2, \ldots, 2)$ be an array of length $\lfloor 10n/3 \rfloor$.
2. Repeat n times:

   Multiply by 10*:* Multiply each entry of A by 10.

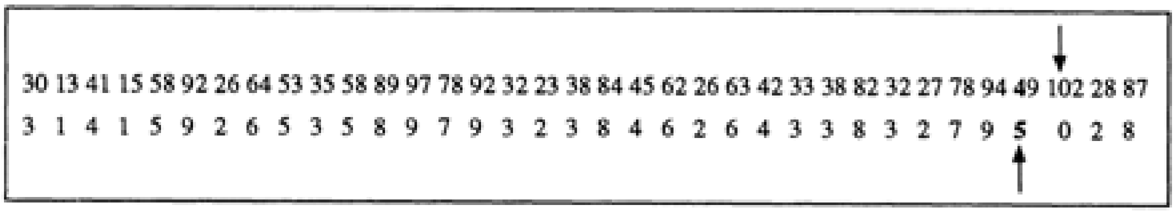
   Put A into regular form: Starting from the right, reduce the ith element of A (corresponding to $\mathbf{c}$-entry $(i-1)/(2i-1)$) modulo $2i-1$, to get a quotient q and a remainder r. Leave r in place and carry $q(i-1)$ one place left. The last integer carried (from the position where $i - 1 = 2$) may be as large as 19.

   Get the next predigit: Reduce the leftmost entry of A (which is at most $109[= 9 \cdot 10 + 19]$) modulo 10. The quotient, q, is the new predigit of π, the remainder staying in place.

   Adjust the predigits: If q is neither 9 nor 10, release the held predigits as true digits of π and hold q. If q is 9, add q to the queue of held predigits. If q is 10 then:
   - set the current predigit to 0 and hold it;
   - increase all other held predigits by 1 (9 becomes 0);
   - release as true digits of π all but the current held predigit.

This algorithm uses only integer arithmetic and is easy to program. The table at the beginning of the paper shows it in action, starting with 13 mixed-radix digits of π (good for 4 base-10 digits). To clarify the working of the algorithm, note that the (finite) first row of Table 1 is a mixed-radix representation of $3.1414796\ldots$, the second row represents $31.414796\ldots$, the fifth row represents $1.414796\ldots$, the sixth row is $14.14796\ldots$, the ninth row is $4.14796\ldots$, and so on. Table 3 shows the result of a computation using a larger initial array; the holding aspect does not become relevant until the 32nd digit.

TABLE 3. The actual digits of π (bottom) compared to the sequence of leftmost base-$\mathbf{c}$ digits for 35 iterations with a starting array of 116 2s (good for 35 digits). At the 32nd iteration a 102 shows up, yielding a predigit of 10.



30	13	41	15	58	92	26	64	53	35	58	89	97	78	92	32	23	38	84	45	62	26	63	42	33	38	82	32	27	78	94	49	102	28	87
3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3	2	3	8	4	6	2	6	4	3	3	8	3	2	7	9	5	0	2	8

We repeat that the algorithm uses only integer operations. To get 5,000 digits of π requires only integer arithmetic on numbers less than 600,000,000. The algorithm leads naturally to the question of improving it to one that is essentially as simple as *e-spigot*.

Question. Is there a base $\mathbf{d}$ of rationals such that π has a $\mathbf{d}$-representation that is periodic, or an arithmetic progression, and such that a_0 is always the integer part of $(a_0; a_1, a_2, \ldots)_{\mathbf{d}}$?

Gosper [**Gos**, p. 32] has discovered a series for π that brings us tantalizingly close to spigot-perfection:

$$\pi = 3 + \frac{1}{60}8 + \frac{1}{60}\frac{2\cdot 3}{7\cdot 8\cdot 3}13 + \frac{1}{60}\frac{2\cdot 3}{7\cdot 8\cdot 3}\frac{3\cdot 5}{10\cdot 11\cdot 3}18 + \frac{1}{60}\frac{2\cdot 3}{7\cdot 8\cdot 3}\frac{3\cdot 5}{10\cdot 11\cdot 3}\frac{4\cdot 7}{13\cdot 14\cdot 3}23 + \cdots .$$

He obtained this series by using a refinement of the Euler transform on $4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \cdots$. Gosper's series leads to the base $\mathbf{d} = (\frac{1}{60}, \frac{6}{168}, \frac{15}{330}, \frac{28}{546}, \ldots)$, with respect to which π is $(3; 8, 13, 18, \ldots)$. A computation shows that $(0; 59, 167, 329, 545, \ldots)_{\mathbf{d}} = 1.092\ldots$, a substantial improvement over the 2 that arose for $\mathbf{c}$. Under the usual randomness assumption for π's digits, the odds of a bad predigit in base $\mathbf{c}$ are 1 in 20, while in base $\mathbf{d}$ they decrease to less than 1 in 110; this is because a $\mathbf{d}$-predigit of 10 occurs only when the remainder is a 9 (which becomes 90) and the carry is a 10. The former happens 10% of the time, while the latter happens no more than once in 11 iterations because the carry is the integer part of a real between 0 and 10.93. So base $\mathbf{d}$ is within 1% of spigot-perfection. Because Gosper's series converges more quickly than the one we used, it has less memory requirements: n digits of π require an initial array of length n; however, the arithmetic on the array will involve integers larger than those in an array of the same size using base $\mathbf{c}$.

One way to improve the Gosper-series approach is to reduce the fractions in $\mathbf{d}$ to lowest terms. Then the regular number with maximal digits is $(0; 59, 27, 21, 38, \ldots)_{\mathbf{d}}$, which equals $1.0000476468\ldots$. It is not hard to see that the regular representation of π is unchanged in this new base. However, the work expended in reducing to lowest terms outweighs the gain made in reducing the number of times a 10 appears as a predigit. Thus it is likely that an affirmative answer to the question above is of more theoretical than practical interest.

The spigot algorithm for π is by no means competitive with the recently discovered fast algorithms (due to the Borwein brothers, the Chudnovsky brothers, and others) that have been used to compute hundreds of millions of digits of π (see [**BBB**]). But the spigot algorithm does have the advantage of avoiding all floating-point computations; thus it is easily implemented on a home computer where it can produce thousands of digits in a few minutes. Moreover, it gives the result directly in base 10 (most other π-algorithms produce the result in binary or some internal format and a second pass must be made to obtain decimal digits).

The algorithm given here can be made to run faster by outputting multiple digits at a time. For example, to get five decimal digits at a time, simply compute the digits of π using base 100,000. This can be done by multiplying by 100,000 instead of 10 in the main step. The integer part is then the next "digit" in base 100,000.

If one is working in base 100,000 and knows in advance that the portion of digits to be computed does not contain the string 00000, then one can omit the lengthy part of the algorithm that adjusts the predigits. This can lead to an exceedingly short computer program. For example, Rabinowitz [**Rab**] used this idea to exhibit a 14-line Fortran program that outputs 1,000 decimal digits of π.

Finally, we mention that the algorithm can be parallelized, in which case it becomes blindingly fast up to about 10,000 digits.

For examples of spigot algorithms for other functions, see [**Abd**].

APPENDIX 1. FIVE LEMMAS

Lemma 1

(a). *If* $i \geq 1$, $(0; 0, 0, \ldots, 0, a_i, a_{i+1}, \ldots)_{\mathbf{b}} < \frac{1}{i}$; *in particular,* $(0; a_1, a_2, a_3, a_4, \ldots)_{\mathbf{b}} < 1$.
(b). *Representations using mixed-radix base* $\mathbf{b}$ *are unique.*
(c). *The integer part of* $(a_0; a_1, a_2, a_3, a_4, \ldots)_{\mathbf{b}}$ *is* a_0 *and the fractional part is* $(0; a_1, a_2, a_3, a_4, \ldots)_{\mathbf{b}}$.

Proof: (a). It suffices to prove that $\sum_{k=i+1}^{\infty}(k-1)/k! = 1/i!$, which follows from the fact that the series telescopes to:

$$\left(\frac{1}{i!} - \frac{1}{(i+1)!}\right) + \left(\frac{1}{(i+1)!} - \frac{1}{(i+2)!}\right) + \left(\frac{1}{(i+2)!} - \frac{1}{(i+3)!}\right) + \cdots .$$

(b). Suppose $(a_0; a_1, a_2, a_3, a_4, \ldots)_{\mathbf{b}}$ and $(c_0; c_1, c_2, c_3, c_4, \ldots)_{\mathbf{b}}$ represent the same real number. Then, for some i, $0 = \sum_{k=i}^{\infty} d_k/k!$, where $|d_k| < k$ and $d_i \neq 0$. But then $|d_i|/i! \leq \sum_{k=i+1}^{\infty} |d_k|/k!$, contradicting (a).

(c). This follows from (a).

Lemma 2. *A positive number is rational iff its digits using the mixed-radix base* $\mathbf{b}$ *are eventually* 0.

Proof: The reverse direction is obvious. For the forward direction we use a sublemma.

Sublemma. *For any integers* t *and* n, *with* $0 \leq n < t!$, *there are integers* d_i *in* $[0, i]$ *such that* $n = d_1 t(t-1)(t-2)\cdots 4\cdot 3 + d_2 t(t-1)(t-2)\cdots 5\cdot 4 + \cdots + d_{t-3}t(t-1) + d_{t-2}t + d_{t-1}$.

Proof: By induction on t. If $n < t!$ write n as $qt + r$ with $0 \leq r < t$ and $0 \leq q < (t-1)!$. By induction there is a sequence $(d_1, d_2, \ldots d_{t-3}, d_{t-2})$ that is a solution for q with respect to terms $(t-1)(t-2)\cdots 4\cdot 3$, and the like, whence $(d_1, d_2, \ldots d_{t-3}, d_{t-2}, r)$ is a solution for n w.r.t. the terms $t(t-1)(t-2)\cdots 4\cdot 3$, and the like.

Returning to Lemma 2's proof, suppose a positive rational s/t is given. Use the sublemma to express $s(t-1)!$ in the form $d_1 t(t-1)(t-2)\cdots 4\cdot 3 + d_2 t(t-1)(t-2)\cdots 5\cdot$

$4 + \cdots + d_{t-3}t(t-1) + d_{t-2}t + d_{t-1}$. Dividing by $t!$ then yields a representation of s/t as a sum of reciprocals of factorials with appropriately small coefficients, which is the same as a terminating representation in the mixed-radix base $\mathbf{b}$.

Lemma 3. *The algorithm for digits of e is correct.*

Proof: It must be shown that $n+2$ mixed-radix digits of e suffice to get n base-10 digits of e. We first prove that if $n \geq 28$ ($= \lceil 10e \rceil$), then n mixed-radix digits suffice for n base-10 digits. Using n mixed-radix digits means we are actually getting the base-10 digits of $e_n = (2; 1, 1, 1, \ldots, 1) = \sum_{i=0}^{n} 1/i!$. Thus we must show that $e - e_n \leq 5 \cdot 10^{-n}$ (see footnote at beginning of paper). A geometric series estimation of the tail of the series shows that $e - e_n < 2/(n+1)!$, and then Stirling's formula yields

$$\frac{2}{(n+1)!} < \frac{1}{n!} < \left(\frac{e}{n}\right)^n < \left(\frac{1}{10}\right)^n.$$

If $n < 28$ then a direct computation of the digits shows that $n+2$ mixed-radix digits suffice.

Lemma 4. *The base-$\mathbf{c}$ number with maximal digits, $(0; 2, 4, 6, 8, \ldots)$, represents 2; hence regular representations of the form $(0; a, b, c, \ldots)_{\mathbf{c}}$ lie between 0 and 2.*

Proof: Instead of giving a formal proof, we show how some *Mathematica* computations led to the result (and a proof). In terms of series, the lemma states that

$$\sum_{i=0}^{\infty} \frac{(2i)i!}{(2i+1)!!} = 2.$$

A rough calculation showed that the sum is near 2. Then a rational computation of the remainders — the differences between the partial sums and 2 — yielded the following sequence.

$$\frac{4}{3}, \frac{4}{5}, \frac{16}{35}, \frac{16}{63}, \frac{32}{231}, \frac{32}{429}, \frac{256}{6435}, \frac{256}{12155}, \frac{512}{46189}, \frac{512}{88179}.$$

The pattern in these remainders was found by dividing each by the preceding one, which yielded:

$$\frac{3}{5}, \frac{4}{7}, \frac{5}{9}, \frac{6}{11}, \frac{7}{13}, \frac{8}{15}, \frac{9}{17}, \frac{10}{19}, \frac{11}{21}.$$

Induction proves the pattern to be valid in general; it follows that the remainders have the closed form $2^{n+1}/\binom{2n+1}{n}$, which converges to 0, as claimed.

Lemma 5. *The algorithm for digits of π is correct.*

Proof: As for e, we look at $\pi - \pi_m$, where $\pi_m = (2; 2, 2, \ldots, 2)_{\mathbf{c}}$. This error is the tail of our main series for π: $\sum_{i=m}^{\infty} (i!)^2 2^{i+1}/(2i+1)!$. This tail is less than twice its first term since each subsequent term is less than half its predecessor, leading us to study $m!^2 2^{m+2}/(2m+1)!$. Splitting the denominator into evens and odds turns this into: $m! 2^2/(3 \cdot 5 \cdots (2m+1))$, which is less than $\frac{2}{3}m!2^2/(2 \cdot 4 \cdots (2m))$, or $1/(3 \cdot 2^{m-1})$. It is easy to see (using the fact that $\frac{3}{10} < \log_{10} 2$) that this last is less than $5 \cdot 10^{-n}$ when $m = \lfloor 10n/3 \rfloor$, as claimed.

APPENDIX 2. PASCAL CODE

The following program, for which we are grateful to Macalester student Simeon Simeonov, implements the algorithm *π-spigot*. This code makes use of the fact that the queue of predigits always has a pile of 9s to the right of its leftmost member, and so only this leftmost predigit and the number of 9s need be remembered. The program computes 1000 digits of π and requires a version of Pascal with a longint data type (32-bit integer).

```
Program Pi_Spigot;
const n      = 1000;
len          = 10*n div 3;
var   i, j, k, q, x, nines, predigit : integer;
      a : array[1..len] of longint;
begin
  for j := 1 to len do a[j] := 2;        {Start with 2s}
  nines := 0; predigit := 0      {First predigit is a 0}
  for j := 1 to n do
  begin q := 0;
    for i := len downto 1 do            {Work backwards}
    begin
      x := 10*a[i] + q*i;
      a[i] := x mod (2*i-1);
      q := x div (2*i-1);
    end;
    a[1] := q mod 10; q := q div 10;
    if q = 9 then nines := nines + 1
    else if q = 10 then
         begin write(predigit+1);
           for k := 1 to nines do write(0);      {zeros}
           predigit := 0; nines := 0
         end
         else begin
           write(predigit); predigit := q;
           if nines <> 0 then
           begin
             for k := 1 to nines do write(9);
             nines := 0
           end
         end
  end;
  writeln(predigit);
end.
```

ADDED IN PROOF. The latest version of *Mathematica* (2.3) can sum many of the series that occur in this paper. It takes only a second or so to get $\pi/2$ as the sum of the crucial series at the beginning of section 2, to get $1/i!$ for the series in Lemma 1's proof, and to get 2 as the sum of the series in Lemma 4's proof.

REFERENCES

[Abd] S. Kamal Abdali, Algorithm 393 — Special series summation with arbitrary precision, *Comm. ACM* **13** (1970) 570.

[BBB] 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, this *Monthly* **96** (1989) 201–219

[Gos] R. W. Gosper, Acceleration of series, Memo no. 304, M.I.T. Artificial Intelligence Laboratory, Cambridge, Mass., 1974.

[Knu] D. E. Knuth, *The Art of Computer Programming,* volume 2, Reading, Mass., Addison-Wesley, 1981.

[Li] J. C. R. Li, Problem E854, this *Monthly* **56** (1949) 633–635.

[Rab] S. Rabinowitz, Abstract 863-11-482: A spigot algorithm for pi, *Abstracts Amer. Math. Society* **12** (1991) 30.

[Sale] A. H. J. Sale, The calculation of e to many significant digits, *Comput. J.* **11** (1968) 229–230.

MathPro Press
P.O. Box 713
Westford, MA 01886
72717.3515@compuserve.com

Department of Mathematics
Macalester College
St. Paul, MN 55105
wagon@macalstr.edu