

Re: Salamin–Brent algorithm

Source: <http://sci.tech–archive.net/Archive/sci.math/2007–06/msg04071.html>

- *From:* dgoldsmith_89 <d.l.goldsmith@xxxxxxxxxx>
 - *Date:* Thu, 21 Jun 2007 22:40:16 –0700
-

On Jun 21, 6:00 pm, Chip Eastham <hardm...@xxxxxxxxxx> wrote:

On Jun 21, 6:42 pm, Gerry Myerson <g...@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

In article <1182447514.837296.169...@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>,

dgoldsmith_89 <d.l.goldsm...@xxxxxxxxxx> wrote:

if I can just download say like
the first 2^{100} binary digits (or 16^{25} hex....

Um, 2^{100} digits – where would you put them all? Seriously.

Anyway, no one has computed more than a few billion digits
(or bits, or whatever), which is way less than what you want.

Well, I agree with the thought (no one would have
room for 2^{100} bits), but actually the computation
of pi has been carried out to one trillion places
(10^{12}) in both decimal and hexadecimal arithmetic.

http://www.super–computing.org/pi_current.html

Thanks for the reference, that looks like it may be useful.

I note that it mentions they used a different algorithm – Divide &
Rationalize; I haven't had a chance to study that, but I have had a
chance to think about the problem of precision and I think I can
phrase my question a little more clearly.

Re: Salamin–Brent algorithm

Assume for the sake of argument that I want to use the S–B algorithm as described at http://en.wikipedia.org/wiki/Gauss–Legendre_algorithm; this requires addition, subtraction, multiplication, division and root taking. Addition and subtraction aren't a big issue as far as affecting the number of digits one needs to keep around: they never change the maximum number by more than one, positive or negative. Multiplication, division, and root–taking, however, are a different story: division and root–taking (multiplication when we're talking about the fraction–representing digits, which is of course almost all the digits in our calculation) right–shift the digits of the numerator/radicand, meaning that left–most digits which one might have thought one could dispose of because they represent precision achieved a while back in the calculation, may instead have to be kept around. But do they? If so, does this problem only delay one's ability to dispose of them, or must all digits always be used every step of the way (necessitating that one's storage capacity increase exponentially, since one's precision is so increasing with the S–B algorithm)? And if one can eventually dispose of left–most digits, what's the disposal criterion?

Does this make sense, or am I missing something which makes these questions nonsense?

DG

as of Oct. 20, 2005.

regards, chip

.