`irwin_v5`: choice of `Mmax`

J.-F. Burnol 2025-05-16 12:28:30

For the mathematical justification of the actual algorithms see the articles linked-to at the README.md. The aim here is to document the choice of `Mmax`. The explanations given were formerly inside the `irwin_v5.sage` as a somewhat lengthy comment. We format it using TEX math notations.

We estimate how many terms are needed according to `level` l and the targeted final precision (before rounding away the guard bits) `nbbits`. This estimate relies on two things:

- a priori lower bound for the final result,
- a priori upper bounds for the terms of the series.

TODO: revisit explanations next and make them clearer.

The $u_{k;m}$'s are bounded above by b. In fact they are bounded above by $\frac{b}{m+1}$ but we don't have this upper bound for the $v_{k;m}$'s. For the latter the beta's are (a bit) smaller as they use inverse powers of some $n+1$'s not n's.

We always take $l =$ `level` at least 2. The crucial upper bound for the m^{th} term of the series comes from the β_{m+1}'s (**ATTENTION**: these β_{m+1}'s also depend on some number of occurrences parameter, not explicited here in the notation). We can estimate that β_{m+1} is bounded above by $\frac{b^{-(l-1)}+m^{-1}}{b^{(l-1)m}}$ with here l being the level. For $k = 0$ and $d = 1$, we can improve that to $\frac{\frac{1}{2}b^{-(l-1)}+m^{-1}}{2^m \cdot b^{(l-1)m}}$.

TODO: fill-in the details and check.

We know from Farhi's theorem that $b\log(b)$ is a lower bound of total sum S (i.e. here S refers not only to the Burnol series but includes the initial, main, contributions) if $k > 0$ or if $k = 0$ and $d = 0$. If $k = 0$ and $d > 0$ we have Proposition 6 in the Irwin paper which says $S > b\log(b) - b\log(1 + 1/d)$. Worst case is with $d = 1$, giving $b\log(b/2)$. For d at least 2 the lower bound improves to $b\log(2b/3)$. Using only that the $u_{k;m}$'s and $v_{k;m}$'s are bounded above by b, we will need to compare $\log(b/2)$ with $((2b^{l-1})^{-1} + m^{-1})2^{-m}$, and $\log(2b/3)$ with $b^{-(l-1)} + m^{-1}$. We always take l at least 2. So it is $\log(b/2)$ versus $(1/(2b) + 1/m)/2^m$ or $\log(2b/3)$ versus $1/b + 1/m$.

Testing the worst case $m = 1$, we have $\log(b/2) > (1/(2b) + 1)/2$ starting at $b = 4$, and $\log(2b/3) > 1/b + 1$ for b at least 5.

For $m = 2$, we have $\log(b/2) > (1/(2b) + 1/2)/4$ for b at least 3 and $\log(2b/3) > 1/b + 1/2$ for b at least 4. But for $b = 3$, the difficulty with $m = 2$ is for $k = 0$ and d neither 0 (as $\log(3) > 1 > 1/3 + 1/2$) or 1, hence $d = 2$. One finds numerically $S > 2.682$, hence in that case $S/3 > 0.894 > 1/3 + 1/2$ indeed.

Only remains to check for $b = 2$. The only Irwin sum not $> 2\log(2)$ is with $k = 0$ and $d = 1$. This is the empty sum with value zero. We do not care too much about our estimates then, use `irwin(2,1,0)` or `irwinpos(2,1,0)` at your own risk... but it seems to work fine.

For `level` $= 2$, β_{m+1} for $b = 2$ is bounded above by $1/2^{m+1} + 1/3^{m+1}$ and $u_{k;m}$ by $2/(m + 1)$. We want to compare this with $2\log(2)/2^m$. It is less already for $m = 1$.

For the positive series we have $2(1/3^{m+1} + 1/4^{m+1})$ to compare with $2\log(2)/2^m$. Again it is less already for $m = 1$.

For `level` > 2 and the alternating series a more precise upper bound of the m^{th} term is $(1/b^{l-1} + 1/m)\frac{b}{m+1}$ times $b^{-(l-1)m}$ so we compare $(1/4 + 1/m)/(m + 1)$ with $\log(2)$. And already for $m = 1$ it is smaller. Still for $b = 2$ and $l > 2$, for the positive series we can bound above the m^{th} term by $(1/(b^{l-1}+1)+1/m)\cdot(b^{l-1}/(b^{l-1}+1))^m b$ times $b^{-(l-1}m)$ using only $v_{k;m} <= b$. And we need to compare with $2\log(2)$. So we check if $(1/5 + 1/m)(4/5)^m$ is less than $\log(2)$. This is true for m at least 2.

In conclusion it is always true that for m at least 2 the m^{th} term of the series contributes a fraction less than $b^{-(l-1)m}$ of the Kempner-Irwin value (we could make a detailed examination for $m = 1$, but drop it). For the alternative series this gives a bound for the total error made by neglecting all terms starting with the m^{th} one; for the positive series, we have to take into account an extra factor of $b^{l-1}/(b^{l-1} - 1)$ which is at most 2. We don't worry about this 2.

So, we only need to take into account the m^{th} term of the series with a precision equal to `nbbits` $-(l-1)m\log(b, 2)$ where `nbbits` is the "full" precision (inclusive of guard bits). We choose the number of terms `Mmax` such that it is the largest integer such that `nbbits` $-(l-1)\cdot$ `Mmax` $\cdot \log(b, 2) \geq$ `nbguardbits`$/2$.

Hence it is defined as (using notation `_Mmax` in case user wants to pass custom choice as `Mmax`):

```
_Mmax = floor((nbbits - nbguardbits/2)/(level-1)/log(b,2))
```

We use precisions of the type `nbbits` $- jT$ with $T =$ `PrecStep`. We will have the need only for j's with `nbbits` $- jT \geq$ `nbguardbits`$/2$ and will thus use

```
NbOfPrec = 1 + floor((nbbits - nbguardbits/2)/PrecStep)
```

distinct precisions. For the actual mapping of index m to the integer j see the code comments in `irwin_v5.sage`.

NOTA BENE:

In the case $d = b - 1$, there is an additional factor $((b - 2)/(b - 1))^m$ coming from $u_{0;m}$ which is not taken into account here (we used only estimates for β_{m+1}'s basically), so we end up using more terms than needed in this case. For example at `level` $= 2$, computing the classic Kempner sum for 500 digits will use about 500 terms but 475 would have been enough as $(8/9)^{475}$ is about 5×10^{-25}. For $k > 0$ and increasing, the effect diminishes. (One can use `verbose=True` to examine the size of the smallest kept term).

For the positive series, there is no such factor but the β_{m+1} is smaller for `level` $=$ 2 roughly by $(b/(b+1))^m$ (for $d = 1$ this will be rather with $b/(b+1/2)$, and $d = 0$ also has another ratio) so we have a similar phenomenon of using too many terms but for other reasons.

For `level` $= 3$ (which is the default) effect will be lower than for `level` $= 2$.

With $k = 0$ (only) and excluded digit 1, we use way too many terms due to roughly a 2^{-m} factor not being taken into account. User can set `Mmax` explicitly to their own choosing.

Here we have chosen `Mmax` according to an analysis which is supposed to work for all k's and d's and `level` > 1.