# SUMALT – Summation of Infinite Alternating Series




**Abstract**

*SUMALT is a program written in 1979 for the HP-34C programmable calculator to quickly and accurately find the sum of infinite alternating series, even divergent ones (Euler sum). Three worked examples are included.*


***Keywords:*** *sum, infinite alternating series, Euler Transformation, Euler sum, differences, programmable calculator, RPN, HP-34C*

## 1. Introduction

*SUMALT* is a short *(84 steps) RPN* program that I wrote in 1979 for the *HP-34C* calculator (will also run *as-is* or with minor modifications in many *RPN* models, such as the *HP-11C*) which, given an infinite alternating series (i.e.: consecutive terms alternate signs) whose general term is defined by the user, it will compute its sum very quickly using the *Euler Transformation* up to 7th-order differences.

The program computes the sum ***S*** of a general infinite series:

$$\boldsymbol{S} = y(0) - y(1) + y(2) - y(3) + \ldots = \sum_{i=0}^{\infty} (-1)^i \, y(i) \qquad i = 0, 1, 2, \ldots$$

and it's most useful when the series converges *very slowly* to its limit sum, as for instance the series:

$$\boldsymbol{S} = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \ldots = Ln\ 2$$

Adding up to about 2000 terms would give only 3 correct places; getting 7 places would need millions of terms and it would take very long, increasing the accumulated error. On the other hand, this program will give the sum correct to 10 places in less than 1 min. It uses the *Euler Transformation*, replacing the original series by this one:

$$\boldsymbol{S} = \frac{1}{2} y(0) - \frac{1}{4} \Delta\, y(0) + \frac{1}{8} \Delta^2\, y(0) - \ldots$$

where the $\Delta^n y(0)$ are the nth-order *differences* of *y(i)*, and the program will use up to 7th-order differences. This procedure is particularly effective for very slowly converging series and is applied not to the original series itself but to the result of subtracting the sum of its first ***n*** terms, where ***n*** is selected by the user.

The procedure goes like this: first, the user defines the series' *general term* under label **B** *(35 steps max.)*, and then the program computes ***S'***, which is the sum of the first ***n*** terms (***n*** is user-specified), and forms a *differences table*, computing differences up the the ***d*** *th* order ($1 \le \boldsymbol{d} \le 7$):

$$\begin{array}{lllll}
y(n+1) & & & & \\
 & \Delta\, y(n+1) & & & \\
y(n+2) & & \Delta^{2} y(n+1) & & \\
 & \Delta\, y(n+2) & \ldots & \Delta^{3} y(n+1) & \ldots \\
y(n+3) & \ldots & & & \\
\ldots & & & &
\end{array}$$

Now *Euler Transformation* is applied, which gives:

$$\boldsymbol{S''} = \frac{1}{2} y(n+1) - \frac{1}{4} \Delta\, y(n+1) + \frac{1}{8} \Delta^2\, y(n+1) - \ldots$$

and the original sum ***S*** is then: $\boldsymbol{S} = \boldsymbol{S'} + \boldsymbol{S''}$.

## 2. Program Listing

| | | | | |
|---|---|---|---|---|
| 01 ♦LBL A | 18 ISG | 35 STO (i) | 52 STO .2 | 69 RCL (i) |
| 02 CF 0 | 19 FIX 4 | 36 ISG | 53 ♦LBL 0 | 70 RCL 8 |
| 03 STO .1 | 20 RCL .0 | 37 FIX 4 | 54 RCL (i) | 71 ÷ |
| 04 X↔Y | 21 RCL I | 38 1 | 55 ISG | 72 STO+ 9 |
| 05 STO .0 | 22 X<=Y | 39 STO+ 8 | 56 FIX 4 | 73 2 |
| 06 1 | 23 GTO 1 ► | 40 RCL .1 | 57 STO- (i) | 74 CHS |
| 07 STO 8 | 24 STO 8 | 41 RCL I | 58 RCL .1 | 75 STOx 8 |
| 08 0 | 25 2 | 42 X<=Y | 59 RCL I | 76 ISG |
| 09 STO 9 | 26 ÷ | 43 GTO 2 ► | 60 X≠Y | 77 FIX 4 |
| 10 STO I | 27 FRAC | 44 2 | 61 GTO 0 ► | 78 RCL .1 |
| 11 ♦LBL 1 | 28 X≠0 | 45 F? 0 | 62 RCL .2 | 79 RCL I |
| 12 GSB B ► | 29 SF 0 | 46 CHS | 63 STO I | 80 X<=Y |
| 13 RCL 8 | 30 CLX | 47 STO 8 | 64 X=0 | 81 GTO 4 ► |
| 14 STO- 8 | 31 STO I | 48 ABS | 65 GTO 4 ► | 82 RCL 9 |
| 15 STO- 8 | 32 ♦LBL 2 | 49 ♦LBL 6 | 66 1 | 83 RTN |
| 16 x | 33 RCL 8 | 50 - | 67 GTO 6 ► | 84 ♦LBL B |
| 17 STO+ 9 | 34 GSB B ► | 51 STO I | 68 ♦LBL 4 | |

- *84 steps*
- *uses registers $R_0$-$R_{.2}$ , $R_I$*
- *uses flag 0*
- *define the general term under ♦LBL B , 35 steps max.*
- *you can use register $R_{.3}$ and up in your definition*
- *the symbols ♦ and ► are purely cosmetic, to indicate branching*

## 3. Usage Instructions

*Step 1:* In **PRGM** Mode, define under 84 ♦LBL B the sequence of steps (35 maximum) which defines the series' *general term,* ***y(i),*** where ***i*** is in stack register *X*, and end it with **RTN**. The very first term corresponds to ***i = 0.*** Do *not* define a sign for each term, it's assumed that it alternates between **+** and **–** .

Also, before keying in the general term's definition do not forget to delete the previous definition from program memory, if there's one, except for 84 ♦LBL B itself.

*Step 2:* In **RUN** Mode, enter the number of terms to sum initially, ***n*** (integer ≥ 0), and the maximum order of differences to compute, ***d*** (integer, $1 \le d \le 7$):

***n*** **ENTER↑** ***d*** **A** ***S*** *(sum of the series)*

To try different values for ***n*** and/or ***d***, repeat *Step 2* above. To sum another series, go to *Step 1* above.

**Notes:**
- the values ***n*** = 7 and ***d*** = 7 are recommended for accuracy and speed, but ***n*** can be > 7 , say ***n*** = 10
- both accuracy and running time depend on ***n*** and ***d***
- *Euler Transformation* is most effective with very slowly convergent series, even *divergent* series can be treated this way and the formal result obtained is then called *Euler Sum* of the divergent series.

## 4. Examples

The following examples can be useful to check that the program is correctly entered and to understand its usage.

### *4.1 Example 1*

Sum the infinite alternating series $S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \ldots$

First of all, we define the general term, which is $y(i) = \frac{1}{i+1}$ : $i = 0, 1, 2, \ldots$

In **RUN** Mode, **GTO B**, switch to **PRGM** Mode and press **1** **+** **1/X** **RTN**, then switch back to **RUN** Mode.

We'll sum the first 10 terms (***n*** = 10), and compute up to 7th-order differences (***d*** = 7) :

**FIX 9** 10 **ENTER** 7 **A** <u>*0.693147182*</u> *(the exact value is Ln 2 = **0.693147181** so we got ~9 correct places)*

*4.2 Example 2*

Evaluate $\boldsymbol{S} = \frac{1}{2}\int_0^1\int_0^{\frac{\pi}{2}}\frac{d\theta\ dk}{\sqrt{1-k^2 Sin^2(\theta)}} = 1-\frac{1}{3^2}+\frac{1}{5^2}-\ldots = \sum_{i=0}^{\infty}(-1)^i\frac{1}{(2i+1)^2}$

First, we define the general term, $y(i) = \frac{1}{(2i+1)^2}$ : *(we assume there are no program steps defined after* `84` $\boxed{\text{♦LBL B}}$*)*

In **RUN** Mode, **GTO B**, switch to **PRGM** Mode and press: **2** **x** **1** **+** **x²** **1/X** **RTN** , then switch back to **RUN** Mode.

We'll sum the first 8 terms (***n*** = 8), and compute up to 7th-order differences (***d*** = 7) :

8 **ENTER** 7 **A** <u>*0.915965595*</u> *(the exact value is **0.915965594**, so we got ~9 correct places again)*

*4.3 Example 3*

Evaluate the *Euler Sum* of the divergent series ***S*** = 1 – 2 + 3 – 4 + 5 – ... $= \sum_{i=0}^{\infty}(-1)^{i}\ (i+1)$

We define the general term, *y(i) = i + 1* : *(we assume there are no program steps defined after* `84` $\boxed{\text{♦LBL B}}$*)*

In **RUN** Mode, **GTO B**, switch to **PRGM** Mode and press: **1** **+** **RTN** , then switch back to **RUN** Mode.

We'll sum no initial terms (***n*** = 0), and compute only the 1st-order differences (***d*** = 1) :

0 **ENTER** 1 **A** <u>*0.250000000*</u>

This is a *divergent* series so it has *no sum*, but consider the function *y(x)* and its *Taylor Series Expansion*:

$$y(x) = \frac{1}{(x+1)^2} = 1 - 2x + 3x^2 - 4x^3 + \ldots$$

The expansion is valid only for *Abs(x) < 1* but if we let *x = 1* anyway both members become:

$$\frac{1}{(1+1)^2} = \frac{1}{4} = \mathbf{0.25} = 1 - 2 + 3 - 4 + \ldots$$

**Notes**

*1.* This program is included in *Hewlett-Packard's Solution Book "HP-34C Matemática Avanzada" (Spanish)*

*2.* The program runs *as-is* in the *HP-11C* and it's featured in my first article *"HP Article VA001 - Long Live the HP-11C"*

**References**

Francis Scheid (1988). *Schaum's Outline of Theory and Problems of Numerical Analysis, 2nd Edition.*
Valentín Albillo (1979). *HP Article VA001 - Long Live the HP-11C*

**Copyrights**