

An approximation of the fine structure constant reciprocal

J. S. Markovitch

P.O. Box 2411, West Brattleboro, VT 05303

Email: jsmarkovitch@yahoo.com

Copyright © J. S. Markovitch, 2005

Abstract

A computer search was made for the most accurate approximation of the fine structure constant

reciprocal $\frac{1}{\alpha}$ in the form $\frac{A^a}{B^b} + C^c$, where the exponents a , b , and c were integers ranging from

0 to 8, and A , B , and C were integers ranging from 1 to 36. Within these restrictions, and

ignoring trivial variants, the best fit required that $A = C = 10$ and $B = 3$, where these integers

prove to be identical to those used earlier by the author in a mass formula.

A computer search was made for most accurate approximation of the fine structure constant reciprocal $\frac{1}{\alpha}$ in the form

$$\frac{A^a}{B^b} + C^c ,$$

where the exponents a , b , and c were integers ranging from 0 to 8, and A , B , and C were integers ranging from 1 to 36. Within these restrictions, and ignoring trivial variants, the best fit was obtained by

$$\frac{1}{\alpha} \approx \frac{10^3}{3^3} + 10^2 = 137.037037... , \quad (1)$$

where the 2002 CODATA value for $\frac{1}{\alpha}$ equals 137.03599911 [1] (see Appendix A for the computer program that uncovered this approximation).

Note that only two distinct integers are needed to produce Eq. (1): so A and C both equal 10, while $B = 3$. Moreover, despite allowing A , B , and C to range as high as 36, both

integers employed are less than 11.

A second computer search was made for most accurate approximation of the fine structure constant reciprocal in the form

$$\frac{10^3 - D^d}{3^3} + 10^2 - E^e ,$$

where the exponents d and e were integers ranging from 0 to 3, and D and E were integers ranging from 1 to 30. Within these restrictions the best fit was obtained by

$$\frac{1}{\alpha} \approx \frac{10^3 - 10^{-3}}{3^3} + 10^2 - 10^{-3} = 137.036 , \quad (2)$$

(See Appendix B for the computer program that uncovered this approximation). Note that the exponents d and e each equal 3, while D and E each equal 10. Moreover, despite allowing D and E to range as high as 30, both integers employed equal 10. So the number 10, which occurred twice in Eq. (1), now occurs twice more in Eq. (2).

It is 10 and 3, along with 4.1, that earlier played a central role in a mass formula introduced by the author [2]. The above approximations should, therefore, be taken as a degree of independent corroboration for a role for 10 and 3 among the fundamental constants of nature.

Acknowledgements

The author wishes to thank Joe Mazur, Eric Ramberg, and C. Y. Lo for their useful comments.

References

- [1] P. J. Mohr, and B. N. Taylor, “The 2002 CODATA Recommended Values of the Fundamental Physical Constants, Web Version 4.0,” available at physics.nist.gov/constants (National Institute of Standards and Technology, Gaithersburg, MD 20899, 9 December 2003).
- [2] J. S. Markovitch, “A Precise, Particle Mass Formula Using Beta-Coefficients From a Higher-Dimensional, Nonsupersymmetric GUT”, available at www.slac.stanford.edu/spires/find/hep/www?r=apri-ph-2003-11 (Applied and Pure Research Institute, Nashua, NH, APRI-PH-2003-11, 2003).

Appendix A

```
//
// FineStructureApproximation.cpp : Find best approximaton of Fine Structure Constant
//

#include "stdafx.h"

#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <limits.h>
#include <ctype.h>
#include <float.h>
#include <errno.h>
#include <signal.h>
#include <locale.h>
#include <setjmp.h>
#include <stdarg.h>

/*-----*/
/*-----*/
/*-----*/

#define MAX_NUMBER 36.0
#define MAX_POWER 8.0

#define AVOID_ROUND_OFF_ERROR 0.0000000001

#define max(x,y)  ( ((x) > (y)) ? (x) : (y) )

/*-----*/
/*-----*/
/*-----*/

int main(int argc, char *argv[])
{
    double dwA, dwAP, dwB, dwBP, dwC, dwCP;
    double dwFSC_Reciprical = 137.03599911; // 2002 CODATA value for FSC
    double dwFSC_Approx = 1000.0/27.0 + 100.0; // value produced by best approximation

    double dwErr1 = fabs(1 - dwFSC_Reciprical / dwFSC_Approx );
    double dwApprox, dwErr2;

    for( dwA = 1.0; dwA <= MAX_NUMBER; dwA++ )
        for( dwAP = 0; dwAP <= MAX_POWER; dwAP++ )

        for( dwB = 1.0; dwB <= MAX_NUMBER; dwB++ )
            for( dwBP = 0; dwBP <= MAX_POWER; dwBP++ )

            for( dwC = 1.0; dwC <= MAX_NUMBER; dwC++ )
                for( dwCP = 0; dwCP <= MAX_POWER; dwCP++ )
                {
                    dwApprox = pow(dwA, dwAP ) / pow(dwB, dwBP ) + pow(dwC, dwCP );

                    dwErr2 = fabs(1 - dwFSC_Reciprical / dwApprox );

                    if( dwErr2 <= dwErr1 + AVOID_ROUND_OFF_ERROR )
                    {
                        printf(

                            "%12.8f = %02.0f^%1.0f / %02.0f^%1.0f + %02.0f^%1.0f (Max=%02.0f)

\n",

                            (float) dwApprox,

                            (float) dwA,
```

```

        (float) dwAP,
        (float) dwB,
        (float) dwBP,

        (float) dwC,
        (float) dwCP,

        (float) max( dwA, max( dwB, dwC ) )
    );
}
}
return 1;
}

```

```

#ifdef RESULTS
137.03703704 = 10^3 / 03^3 + 10^2 (Max=10)
137.03703704 = 10^3 / 27^1 + 10^2 (Max=27)
137.03703704 = 10^6 / 30^3 + 10^2 (Max=30)
137.03703704 = 20^3 / 06^3 + 10^2 (Max=20)
137.03703704 = 30^3 / 03^6 + 10^2 (Max=30)
137.03703704 = 30^3 / 09^3 + 10^2 (Max=30)
137.03703704 = 30^3 / 27^2 + 10^2 (Max=30)
Press any key to continue

```

```

#endif

```

Appendix B

```
//
// FineStructureApproximationRefinement.cpp : Find best approximat on of Fine Structure Constant
//

#include "stdafx.h"

#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <limits.h>
#include <ctype.h>
#include <float.h>
#include <errno.h>
#include <signal.h>
#include <locale.h>
#include <setjmp.h>
#include <stdarg.h>

/*-----*/
/*-----*/
/*-----*/

#define MAX_NUMBER 30.0
#define MAX_POWER 3.0

#define AVOID_ROUND_OFF_ERROR 0.0000000001

/*-----*/
/*-----*/
/*-----*/

int main(int argc, char *argv[])
{
    double dwD, dwDP, dwE, dwEP;
    double dwFSC_Reciprical = 137.03599911; // 2002 CODATA value for FSC
    double dwFSC_Approx = 137.036; // value produced by best approximation

    double dwErr1 = fabs(1 - dwFSC_Reciprical / dwFSC_Approx );
    double dwApprox, dwErr2;

    for( dwD = 1.0; dwD <= MAX_NUMBER; dwD++ )
        for( dwDP = 0; dwDP <= MAX_POWER; dwDP++ )

            for( dwE = 1.0; dwE <= MAX_NUMBER; dwE++ )
                for( dwEP = 0; dwEP <= MAX_POWER; dwEP++ )
                {

                    dwApprox = (pow(10.0, 3.0) - pow(dwD, -dwDP)) / pow(3.0, 3.0)
                        + pow(10.0, 2.0) - pow(dwE, -dwEP);

                    dwErr2 = fabs(1 - dwFSC_Reciprical / dwApprox );

                    if( dwErr2 <= dwErr1 + AVOID_ROUND_OFF_ERROR )
                    {
                        printf(

                            "%12.8f = (%02.0f^%1.0f - %02.0f^-%1.0f) / %02.0f^%1.0f + %02.0f^%1.0f - %02.0f^-%1.0f\n",

                            (float) dwApprox,

                            (float) 10,
                            (float) 3,
```

```
        (float) dwD,  
        (float) dwDP,  
  
        (float) 3,  
        (float) 3,  
  
        (float) 10,  
        (float) 2,  
  
        (float) dwE,  
        (float) dwEP  
    );  
    }  
    return 1;  
}  
  
#ifdef RESULTS  
137.03600000 = (10^3 - 10^-3) / 03^3 + 10^2 - 10^-3  
#endif
```