# HP Forum Archive 16

[ Return to Index | Top of Index ]

## Numerical integration on the 11C

*Message #1 Posted by **Gerson W. Barbosa** on 16 June 2006, 11:34 p.m.*

This program was written in 1985, originally for the HP-15C. Of course this was not needed on the 15C, since its built-in integrator is way better than Simpson's 1/3 method. By what I can remember, I was just trying the algorithm on the 15C before writing a PASCAL program for my Numerical Analysis homework (No computer at home then).

This would have been more useful back then but here it is just the same:

```
-----------------------------------------------------


Numerical Integration on the HP-11C (Simpson's Method)



f LBL A          | 001-42,21,11 |
STO I            | 002-   44 25 |
Rv               | 003-      33 |
STO 2            | 004-   44  2 |
Rv               | 005-      33 |
STO 1            | 006-   44  1 |
CHS              | 007-      16 |
RCL 2            | 008-   45  2 |
+                | 009-      40 |
RCL I            | 010-   45 25 |
/                | 011-      10 |
STO 4            | 012-   44  4 |
0                | 013-       0 |
STO 3            | 014-   44  3 |
2                | 015-       2 |
RCL I            | 016-   45 25 |
f x=y            | 017-   42 40 |
GTO 2            | 018-   22  2 |
2                | 019-       2 |
```

```
|-              | 020-      30 |
|2              | 021-       2 |
|/              | 022-      10 |
|5              | 023-       5 |
|CHS            | 024-      16 |
|10^x           | 025-      13 |
|+              | 026-      40 |
|STO I          | 027-   44 25 |
|1              | 028-       1 |
|STO 5          | 029-   44  5 |
|f LBL 1        | 030-42,21, 1 |
|RCL 4          | 031-   45  4 |
|RCL 5          | 032-   45  5 |
|*              | 033-      20 |
|2              | 034-       2 |
|*              | 035-      20 |
|RCL 1          | 036-   45  1 |
|+              | 037-      40 |
|GSB 0          | 038-   32  0 |
|2              | 039-       2 |
|*              | 040-      20 |
|STO + 3        | 041-44,40, 3 |
|RCL 5          | 042-   45  5 |
|2              | 043-       2 |
|*              | 044-      20 |
|1              | 045-       1 |
|+              | 046-      40 |
|RCL 4          | 047-   45  4 |
|*              | 048-      20 |
|RCL 1          | 049-   45  1 |
|+              | 050-      40 |
|GSB 0          | 051-   32  0 |
|4              | 052-       4 |
|*              | 053-      20 |
|STO + 3        | 054-44,40, 3 |
|1              | 055-       1 |
|STO + 5        | 056-44,40, 5 |
|f DSE          | 057-   42  5 |
|GTO 1          | 058-   22  1 |
|f LBL 2        | 059-42,21, 2 |
|RCL 1          | 060-   45  1 |
|GSB 0          | 061-   32  0 |
|STO + 3        | 062-44,40, 3 |
|RCL 2          | 063-   45  2 |
|GSB 0          | 064-   32  0 |
|STO + 3        | 065-44,40, 3 |
|RCL 1          | 066-   45  1 |
```

```
RCL 4           | 067-   45  4 |
+               | 068-      40 |
GSB 0           | 069-   32  0 |
4               | 070-       4 |
*               | 071-      20 |
STO + 3         | 072-44,40, 3 |
RCL 3           | 073-   45  3 |
RCL 4           | 074-   45  4 |
3               | 075-       3 |
/               | 076-      10 |
*               | 077-      20 |
g RTN           | 078-   43 32 |
```

Usage:

- write the function to be integrated as subroutine 0;

- enter the lower limit, the upper limit and an even number of intervals on the stack;

- execute A

Example:

$$\int_0^{\wedge 2} \frac{\sin(x)}{x}\,dx$$

```
f LBL 0         | 079-42,21, 0 |
SIN             | 080-      23 |
g LSTx          | 081-   43 36 |
/               | 082-      10 |
g RTN           | 083-   43 32 |
```

```
g RAD
1e-99          (to avoid division by zero);
ENTER
2
ENTER
10
f A



1.605415090     (not so bad, exact value to nine places is 1.605412977)


-----------------------------------------------------
```

## Re: Numerical integration on the 11C

*Message #2 Posted by **Karl Schneider** on 17 June 2006, 1:16 a.m.,*
*in response to message #1 by Gerson W. Barbosa*

Hi, Gerson --

Interesting and useful program. I did not compare it against the combined trapezoidal/Simpson's Rule program in the HP-11C Owner's Handbook (pp. 159-162).

My accelerated HP-11C will produce the answer you gave in less than 10 seconds. The exact answer can be obtained with n between somewhere 80 and 100.

*Note: The acceleration was achieved by connecting a 33 uH inductor in parallel with the LC oscillator circuit, making its resonant frequency 2.5 times as high, making the calculator run on a clock correspondingly faster. The post (which you replied to) is at*

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv015.cgi?read=78272#78272

*No, I still haven't taken photos...*

My 1981 Casio fx-3600P (and 2004 fx-115MS) has built-in Simpson's Rule integration. N is calculated as $2^n$, where n is specified as an integer from 1-9. This guarantees an even N, and allows a wide range of N to be specified easily.

Regards,

-- KS

*Edited: 17 June 2006, 1:27 a.m.*

## Re: Numerical integration on the 11C

*Message #3 Posted by **Gerson W. Barbosa** on 17 June 2006, 1:32 p.m.,*
*in response to message #2 by Karl Schneider*

Hello Karl,

I'm glad you have found it useful. The program in the manual is shorter, but the function has to be evaluated at equally spaced points and the results have to be entered manually which is rather cumbersome.

The program runs in 25 seconds on my 11C for n=10. I tried to speed up a 15C but the batteries would die too fast so I quit. I think the inductor I used had a very low resistance.

The program still lacks an accuracy factor. What we know for sure is that it will give exact results for polynomial functions with degrees up to 2.

Best regards,

Gerson.

## Re: Numerical integration on the 11C

*Message #4 Posted by **Valentin Albillo** on 17 June 2006, 11:57 p.m.,*
*in response to message #1 by Gerson W. Barbosa*

Hi, Gerson:

Nice program, but Simpson's method is to numerical integration what bubble sort is to sorting, i.e., vastly inefficient. The best methods, and the ones who would've been included as the basis of INTEG in the HP-34C (and subsequently HP-15C, HP-71B's Math ROM, etc) are Gaussian ones, were it not for the fact that they do require a certain number of full-precision real constants that would require a noticeable amount of ROM space, not available in the HP-34C project, among other considerations, as discussed in Mr. Kahan's wonderful article on INTEG, originally published in the ***HP Journal*** eons ago.

However, Gaussian methods are still way better than Simpson's rule even for low-point variants. For instance, here's my HP-11C version of a Gaussian integration routine I originally wrote for the HP-41C, as published in ***PPC Journal V7N6 page 10 (ROM Progress section), Jul/Aug 1980***, i.e. exactly 26 years ago:

**Gaussian integration for the HP-11C**

```
01 LBL A     17 LBL 0     32 RCL 1
02 STO 1     18 RCL 1     33 GSB E
03  -        19 RCL 3     34  8
04 RCL I     20  +        35  *
05  /        21 GSB E     36 STO-2
06 STO 0     22  5        37 RCL 0
07  2        23  *        38 STO+1
08  /        24 STO-2     39 DSE
09 STO+1     25 RCL 1     40 GTO 0
10  .6       26 RCL 3     41 RCL 2
12 SQRT      27  -        42  *
13  *        28 GSB E     43 18
14 STO 3     29  5        45  /
15 CLX       30  *        46 RTN
16 STO 2     31 STO-2     47 LBL E
```

As you may see, it's a *very small, 46-step program* which uses just R0-R3 for scratch and RI as a decrementing counter for the number of subintervals. It delivers **exact** results, *even using just 1 subinterval*, for f(x) being a polynomial of degrees up to (and including) **5th**, while evaluating f(x) just 3 times per subinterval. That's about twice as precise as Simpson's rule, which delivers exact results for polynomials up to 3rd degree only (not 2nd, as stated in another post).

To use it to compute the integral of an arbitrary **f(x)** between x=**a** and x=**b**, using N-subinterval Gaussian integration, just do the following:

1. Enter your f(x) to be integrated into program memory, starting at

   ```
   47 LBL E
   ```

   ending it either with a RTN instruction or with the end of program memory.

2. Store **N**, the number of subintervals you want to use, in Register I. **N** must be an integer number equal or greater than 1. The larger N, the more precise the result will be and the longer it will take to run.

   ```
   N,  STO I
   ```

3. Enter the limits of integration, **a** and **b**, into the stack and call the integration routine:

   ```
   a, ENTER, b, GSB A
   ```

The computation will proceed and the result will be displayed upon termination. Let's see a couple of examples:

**1. Compute the integral of f(x) = $6x^5$ between x=-6 and x=45.**

As f(x) is a 5th degree polynomial, we expect an exact result, save for minor rounding errors in the very last place. Let's compute it:

- Define f(x):

```
LBL E, 5, y^x, 6, *, RTN
```

- Store the number of subintervals, just one will do:

```
1, STO I
```

- Enter the limits and compute the integral:

```
6, CHS, ENTER, 45, GSB A
```

- The result is returned within 6 seconds:

-> **8,303,718,967**

The exact integral is **8,303,718,969**, so our result is *exact to 10 digits* within 2 ulps, despite using just *one* subinterval and despite the large interval of integration.

Now for your own example:

**2. Compute the integral of f(x)=Sin(x)/x between x=0 and x=2**

- Set RAD mode and define f(x):

```
LBL E, SIN, LASTX, /, RTN
```

- Store the number of subintervals, let's try just 1:

```
1, STO I
```

- Enter the limits and compute the integral:

```
1E-99, ENTER, 2, GSB A
```

```
- The result is returned within 5 seconds:


    -> 1.605418622
```

Testing with 2,3, and 4 subintervals we get (the exact integral being **1.605412977**):

```
N subintervals  Computed integral  Time
-----------------------------------------
   1             1.605418622        5 sec.
   2             1.605413059       11 sec.
   3             1.605412984       16 sec.
   4             1.605412978       22 sec.
```

so *even using just 2 subintervals does provide 8-digit accuracy*, and using 4 nails down the result to *10 digits* save for a single unit in the last place.

That's all. I think you'll agree this simple Gaussian method beats Simpson's and Trapezoidal Rules hands down. :-)

Best regards from V.

## Re: Numerical integration on the 11C

*Message #5 Posted by **Namir** on 18 June 2006, 8:58 a.m.,*
*in response to message #4 by Valentin Albillo*

Now that's what I am talking about when it comes to sharing cool algorithms and implementations. It was good then in 1980 and its stil good now!!

Namir

### Thanks, Namir ! :-) [no text]

*Message #6 Posted by **Valentin Albillo** on 18 June 2006, 8:02 p.m.,*
*in response to message #5 by Namir*

Best regards from V.

## Re: Numerical integration on the 11C

*Message #7 Posted by **Gerson W. Barbosa** on 18 June 2006, 11:54 a.m.,*
*in response to message #4 by Valentin Albillo*

Hi Valentin,

Thanks for your reply. As I said, I wrote that program only as an exercise. I do know Simpson's method is not the most efficient one, especially for programmable calculators. But for practical engineering use it suffices. I remember one of my Calculus teachers, a civil engineer, once told us how he evaluated an integral in the field: he just drew the function on cardboard, cut the area he wanted to measure, weighted it on a precision scale and compared it with the weight of a square unit of that cardboard. I don't know why he needed that integral for, but I guess if he was building a bridge, it still stands :-) (Please don't get me wrong: I do appreciate exact and clever solutions whenever possible).

Anyway, submitting a 78-step long program and getting a better 46-step one in return was really a big deal. I have just filed your posting in my 11C folder. Thanks again! :-)

Best regards,

Gerson.

## Re: Numerical integration on the 11C

*Message #8 Posted by **Valentin Albillo** on 18 June 2006, 8:26 p.m.,*
*in response to message #7 by Gerson W. Barbosa*

Hi again, Gerson:

Gerson posted:

*"Thanks for your reply. As I said, I wrote that program only as an exercise."*

> Yes, of course, it wasn't my intention to belittle your valuable efforts, it's just that numerical integration (a.k.a. *numerical quadrature*) has been a pet math topic of mine since I can remember, and even with my HP-25 I was already mining info here and there for the most powerful, accurate algorithms. Simpson's rule seemed unacceptably poor to me and that way, 30+ years ago, I discovered Gaussian quadrature.
>
> Gaussian quadrature methods are among the theoretically very best existing algorithms for numerical integration. They accomplish the feat by specifying *both* the precise x-ordinates where the function will be evaluated, as well as the corresponding weights, which results in the N-point Gaussian quadrature being exact for polynomials up to degree 2*N-1, which is approximately *twice as much* as most other simpler methods. This means they provide much more accurate results using the same number of function evaluations per subinterval as other methods, or conversely, they'll achieve the same accuracy with *half* the number of evaluations, thus being much faster.
>
> Their only caveat is the need to store both the x-ordinates and the weights previously computed to the required accuracy, or else, to compute them on the fly for a session, again to the necesssary accuracy. However, this is actually quite simple to implement, and matter of fact I did write a *17-line*

implementation that will compute an N-point Gaussian quadrature for any user-specified function to any given accuracy, repeatedly doubling the number of subintervals till the desired accuracy is met. The user simply specifies the function to be integrated, the desired accuracy, the integration limits and the number N of Gaussian points to use, and that's is.

My program will then compute the N required x-ordinates (which takes only 8 lines of code in all), and then proceeds to perform N-point Gaussian quadrature using 1, 2, 4, 8, ... subintervals (which takes the remaining 9 lines of code) till the desired accuracy is met or exceeded.

For instance, for your very own example, integrating Sin(x)/x between x=0 and x=2, specifying a mere 16-point Gaussian quadrature returns the integral *correct to 48 digits using just *one* subinterval*, and to 57 digits using just two, namely the value:

```
1.605412976802694848576720148198588940848583422328499660 28
```

where the last digit should be a 9. How's that for accuracy (48-57 correct digits) and speed (only 16-32 function evaluations) ?

Even so, standard Gaussian quadrature methods are *not* the cutting edge in numerical quadrature nowadays, there are other advanced, more powerful methods that will cope with just about any integral automatically, even if improper, to any desired accuracy, the caveat being they are much more complex and delicate to correctly implement. But exist, they do.

Best regards from V.

## Some questions: Gaussian vs. HP's modified Romberg

*Message #9 Posted by **Karl Schneider** on 23 June 2006, 2:51 a.m.,*
*in response to message #8 by Valentin Albillo*

Hello, Valentin --

I've done some rudimentary investigation regarding your two fine posts about Gaussian quadrature (numerical integration) in this thread:

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/forum.cgi?read=94530

http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/forum.cgi?read=94606

Wikipedia provides a decent introduction to the techniques of numerical integration:

http://en.wikipedia.org/w/index.php?title=Numerical_integration

Links to entries for Romberg's Method, Gaussian quadrature, Newton-Cotes formulas, and Simpson's Rule can be easily found.

The above article states the following:

*"A Gaussian quadrature rule is typically more accurate than a Newton-Cotes rule which requires the same number of function evaluations, if the integrand is smooth (i.e., if it has many derivatives.)*

*"If f does not have many derivatives at all points, or if the derivatives become large, then Gaussian quadrature is often insufficient."*

Would you fully agree with the above assessments? It certainly seems that HP emphasized robustness in the modified Romberg's method implemnted for the HP-34C (and described in the *HP Journal* article from 1980), in that answers correct to the level specified by the user should be returned even for highly irregular functions. I like the example for

f(x) = sqrt(x)/(x-1) - 1/ln(x) from x = 0 to 1

treated in the paper, the HP-15C Advanced Functions Handbook, and the HP-71B Math ROM manual. This function is undefined at x = 0 and x = 1, and is very "spiky" near x = 0. The returned answer is very accurate for the HP-34C, HP-15C, HP-32S, and HP-32SII as long as "FIX 6" or higher display mode is specified for input function accuracy.

The article about Newton-Cotes formulas states,

*"If the evaluation points are not assumed to be equally spaced, another class of formulas, called Gaussian quadrature formulas, can be derived."*

HP's Romberg implementation uses irregularly-spaced samples, to prevent aliasing of periodic functions from giving incorrect answers. Does this modification make it a Gaussian method?

The link to Gaussian quadrature shows the points and weighting factors for respective numbers of points per division. Now I see where sqrt(0.6), 5/9, and 8/9 came from in your 1980 program. The points and weights for 4 or 5 points per division seem truly arcane and not suited for a compact program.

Even if modern calculator hardware could handle more-sophisticated quadrature methods, one has to admire the practical sophistication of HP's original 1979 implementation for the HP-34C, which has certainly withstood the test of time:

- Providing the user a means of specifying the accuracy of the *input* function
- Estimating the maximum error of the result
- Not requiring the user to input the number of sample points
- Not requiring that the function be evaluable at the end points

Less impressive is the Simpson's Rule technique employed for the keystroke programs in the HP-41 Math Pac, the HP-20S, and low-end Casios. Your Gaussian-quadrature example makes me wonder why it wasn't used for these offerings.

Best regards,

-- KS

## Re: Some questions: Gaussian vs. HP's modified Romberg

*Message #10 Posted by **Valentin Albillo** on 23 June 2006, 11:29 p.m.,*
*in response to message #9 by Karl Schneider*

Hi again, Karl:

Karl posted:

*""If f does not have many derivatives at all points, or if the derivatives become large, then Gaussian quadrature is often insufficient."*

*Would you fully agree with the above assessments? It certainly seems that HP emphasized robustness in the modified Romberg's method implemnted for the HP-34C (and described in the HP Journal article from 1980), in that answers correct to the level specified by the user should be returned even for highly irregular functions.*

> No, I don't fully agree. I've computed many very difficult integrals with high-point Gaussian quadratures, and they normally cope with almost anything. For very difficult integrands, the convergence to an exact result slows down, but except with frankly horrendous, infinitely-oscillating integrands near singularities, I've rarely seen it fail, and even then, you can succeed by using a Gaussian quadrature tailored to the problem at hand by means of a suitable weighting function (such as some trigonometric expression) which will absorb the oscillations and singularities as if nonexistent. That said, as I stated in a previous post, Gaussian quadrature isn't the most advanced, state-of-the-art quadrature method right now, there are better ones.
>
> As for the "robust" HP-34C Romberg's method's property that "answers correct to the level specified by the user should be returned even for highly irregular functions", I've seen lots and lots of integrands failing to meet that desirable goal, and several are pointed out in an issue of HP Journal, where very convoluted, high-math manoeuvres are necessary to deal with them, assuming you can recognize that you're losing precision and that the error estimate is nonchalantly deceiving you, that is.

*"HP's Romberg implementation uses irregularly-spaced samples, to prevent aliasing of periodic functions from giving incorrect answers. Does this modification make it a Gaussian method? "*

> Absolutely not. It still is a low-order method heavily relying in halving, iterating, acceleration, and prediction, which is about the best it can do, though it needs 23 registers and a lot of code to do its thing.

Best regards from V.

## Re: Numerical integration on the 11C

*Message #11 Posted by **Gerson W. Barbosa** on 18 June 2006, 6:41 p.m.,*
*in response to message #4 by Valentin Albillo*

> Quote:
>
> That's about twice as precise as Simpson's rule, which delivers exact results for polynomials up to $3^{rd}$ degree only (not $2^{nd}$, as stated in another post).

My bad! Since in Simpson's rule the function to be integrated is approximated by parabolic arcs (i.e. $2^{nd}$ degree polynomials), it seemed obvious to me it would yield exact results only for polynomial up to $2^{nd}$ degree... but I was wrong. Likewise Simpson's 3/8 Rule will give exact results for polynomials up to $4^{th}$ degree, won't it?

Best regards,

Gerson.

*Edited: 19 June 2006, 8:54 p.m.*

## Re: Numerical integration on the 11C

*Message #12 Posted by **Les Wright** on 19 June 2006, 1:39 a.m.,*
*in response to message #4 by Valentin Albillo*

On a 42S, appropriately modified (i.e., XEQ for GSB), your routine gives 8,303,718,968.94 and with one interval and the exact result for two intervals in the polynomial example.

For the sine integral with 8 intervals I get 1.60541297682.

I didn't put a stop watch to any of this, but the polynomial integration was quick (less than a couple of seconds), and the latter calculation was in the 8-10 sec range tops. I also used 1E-499 as the tiny element there.

I am embarrassed to admit this, but for the first few attempts to compute the sine integral I kept getting about 0.035. You will never guess why....

Neat little program. Thanks for sharing it!

Les

## Re: Numerical integration on the 11C

*Message #13 Posted by **Massimo Gnerucci (Italy)** on 19 June 2006, 1:46 a.m.,*
*in response to message #12 by Les Wright*

> Quote:
>
> I am embarrassed to admit this, but for the first few attempts to compute the sine integral I kept getting about 0.035. You will never guess why....

DEG vs RAD ?

Massimo

## Re: Numerical integration on the 11C

*Message #14 Posted by **Les Wright** on 19 June 2006, 2:18 a.m.,*
*in response to message #13 by Massimo Gnerucci (Italy)*

Yup. Talk about a bush-league error. I was driving myself crazy for a good 20 minutes.... :)

## Re: Numerical integration on the 11C

*Message #15 Posted by **Valentin Albillo** on 19 June 2006, 4:22 a.m.,*
*in response to message #12 by Les Wright*

Hi, Les:

Les posted:

*"On a 42S, appropriately modified (i.e., XEQ for GSB), your routine gives 8,303,718,968.94 and with one interval and the exact result for two intervals in the polynomial example."*

It is theoretically exact even with just one subinterval, unavoidable rounding errors being the culprit for the 0.06 deviation. Also, apart from XEQ for GSB, etc, RCL arithmetic can be used to noticeably shorten the program, as I'm sure you're fully aware. Further improvements are possible in the HP-42S but there's little point as it does include its own built-in integrator. It would be interesting to compare them, my program in a 42S-optimized version versus the built-in integrator.

*"I also used 1E-499 as the tiny element there."*

Actually, there's no need for it, you can enter 0 as the low limit, because Gaussian quadrature doesn't evaluate the function at the limits of integration. I used 1E-99 just to exactly mimic Gerson's example, but the original 0 would be perfectly Ok.

*"Neat little program. Thanks for sharing it!"*

You're welcome, glad you liked it.

Best regards from V.

## Re: Numerical integration on the 11C

*Message #16 Posted by **Antonio Maschio (Italy)** on 20 June 2006, 2:16 a.m.,*
*in response to message #4 by Valentin Albillo*

**Astonishing!**

-- Antonio

## Thank you, Antonio ! (no text)

*Message #17 Posted by **Valentin Albillo** on 20 June 2006, 4:33 a.m.,*
*in response to message #16 by Antonio Maschio (Italy)*

Best regards from V.

## Re: Numerical integration on the 11C

*Message #18 Posted by **Gerson W. Barbosa** on 20 June 2006, 11:03 p.m.,*
*in response to message #4 by Valentin Albillo*

Hi again Valentin,

My example is actually the one in the HP-15C manual, which originally appeared in the HP-34C manual. Though it is quickly evaluated on both calculators (FIX 4), your program is even faster, at least for this particular example. What about this integral? ( CHS $e^x$ LSTx $x^2$ 3 1/x $y^x$ 8 * SIN * 1 + )

I computed it with your program (N=20) and obtained 2.016503557 after 3 minutes and 40 seconds (the best approximation should be 2.016279720). By what I remember it takes more than 10 minutes to be solved on the 32SII. A difficult integral for numerical methods?

Best regards,

Gerson.

## Numerical integration on the 15C and 32SII (was "11C")

*Message #19 Posted by **Karl Schneider** on 21 June 2006, 10:57 p.m.,*
*in response to message #18 by Gerson W. Barbosa*

Hi, Gerson --

> Quote:
>
> What about this integral?

Integral of f(x)dx between 0 and 2, with $f(x) = e^{-x} \sin(8x^{2/3})$

> Quote:
>
> I computed it with your *(Valentin's 1980 Gaussian quadrature)* program (N=20) and obtained 2.016503557 after 3 minutes and 40 seconds (the best approximation should be 2.016279720). By what I remember it takes more than 10 minutes to be solved on the 32SII. A difficult integral for numerical methods?

The nifty animated plot of the function at the linked website (courtesy of California State University at Fullerton), shows that the function actually is quite well-behaved, with no spikes or discontinuities. The function is defined at every point between the limits of 0 and 2, inclusive. The slope is steep between 0 and the first local maximum at about x = 0.083 (which accounts for the error in the Simpson's Rule approximation), but otherwise it is not problematic.

The HP-15C and HP-32SII with their modified Romberg's Method can outperform both Simpson's Rule (integral = 2.015906 with N=120) and the Gaussian quadrature result you obtained (integral = 2.016503557 with N=20).

Using "FIX 5", "FIX 8" and "FIX 9" to define the input-function uncertainty as $5 \times 10^{-6}$, $5 \times 10^{-9}$ and $5 \times 10^{-10}$ respectively, I obtained the following results:

HP-15C:

| Setting | Result | Estimated error | Execution time [sec] |
|---|---|---|---|
| FIX 5 | 2.016278200 | $1.00005 \times 10^{-5}$ | 245 (4:05) |
| FIX 8 | 2.016279718 | $1.05 \times 10^{-8}$ | 1921 (32:01) |

HP-32SII:

| Setting | Result | Estimated error | Execution time [sec] |
|---|---|---|---|
| FIX 5 | 2.01627820014 | $1.0000005 \times 10^{-5}$ | 19 (0:19) |
| FIX 8 | 2.01627971820 | $1.0005 \times 10^{-8}$ | 143 (2:23) |
| FIX 9 | 2.01627971948 | $1.005 \times 10^{-9}$ | 285 (4:45) |

As you can see, I got accurate results on the HP-32SII in well under 10 minutes. The HP-32SII is 12 (or more) times as fast as the HP-15C, and has two more significant digits of numerical representation. Mathematical equations entered as programs will run faster than as "EQN" expressions, but that won't account for such a large difference in time.

I used your program on my HP-15C.

My HP-32SII program:

```
RCL X
+/-
e^x
RCL X
.2.3  (0.666666666667)
y^x
8
*
SIN
*
1
+
RTN
```

Best regards,

-- KS

*Edited: 22 June 2006, 3:58 a.m.*

## Re: Numerical integration on the 15C and 32SII (was "11C")

*Message #20 Posted by **Gerson W. Barbosa** on 22 June 2006, 2:19 p.m.,*
*in response to message #19 by Karl Schneider*

Hi Karl,

Thanks for taking the time to timing all of those! I forgot to mention I was using the 32SII with standard display mode. I just rounded the result to nine places to match the expected result on the 11C. Actually the running time, using an "EQN" expression, was well above 10 minutes. Hoping to get faster results, I tried it on the HP-33S, using a program as you've suggested, but the timings were pretty close to those on the HP-32SII, as you can see in the following table:

HP-33S:

| Setting | Result | Estimated error | Execution time [sec] |
|---|---|---|---|
| FIX 5 | 2.01627820014 | 2.016281 x 10-5 | 18 (0:18) |
| FIX 8 | 2.01627971820 | 2.016280 x 10-8 | 130 (2:10) |
| FIX 9 | 2.01627971948 | 2.016280 x 10-9 | 260 (4:20) |
| ALL | 2.01627971962 | 2.016280 x 10-11 | 1034 (17:14) !!! |

The program:

```
LBL Z
RCL X
+/-
e^x
LASTx
x^2
∛x        ; The cubic root on the 33S should be slightly faster than y^x
*
SIN
*
1
+
RTN
```

With today's technology, I'd like to get it solved in 17 seconds rather than 17 minutes... Well, I think the HP-49G+ can do that :-)

It seems a really complicated integral for numerical methods was presented here some months ago, but I don't remember the thread's subject...

Best regards,

Gerson.

## Re: Numerical integration on the 15C and 32SII (was "11C")

*Message #21 Posted by **Karl Schneider** on 23 June 2006, 12:40 a.m.,*
*in response to message #20 by Gerson W. Barbosa*

Hello, Gerson --

> Quote:
>
> I forgot to mention I was using the 32SII with standard display mode. I just rounded the result to nine places to match the expected result on the 11C.

Ah, that's the issue. That "standard" display mode, with no trailing zeroes displayed, is the only mode on the HP-35 and is the default mode on non-HP algebraic models. The mode is called "ALL" (for all necessary decimal digits) on the Pioneer-series models. For purposes of defining the uncertainty of the integrand function, it's essentially equivalent to "SCI 11". This will cause the most exacting and extensive computation of the integral, resulting in the long times you observed.

There's a distinct pattern in these results -- twice as much time for every extra digit of accuracy.

> Quote:
>
> Hoping to get faster results, I tried it on the HP-33S, using a program as you've suggested, but the timings were pretty close to those on the HP-32SII.

We found several years ago that the HP-33S is only incrementally faster than the HP-32SII -- maybe 10-20%. That's a bit disappointing, but I've always felt that the Saturn processor offered satisfactory performance for any non-graphing calculator lacking the fancy features of the original 28/48/49 lines.

-- KS

*Edited: 23 June 2006, 12:51 a.m.*

## Re: Numerical integration on the 15C and 32SII (was "11C")

*Message #22 Posted by **Valentin Albillo** on 23 June 2006, 11:11 p.m.,*
*in response to message #19 by Karl Schneider*

Hi, Karl:

Karl posted:

*"The HP-15C and HP-32SII with their modified Romberg's Method can outperform both Simpson's Rule (integral = 2.015906 with N=120) and the Gaussian quadrature result you obtained (integral = 2.016503557 with N=20)."*

Your comparison against my Gaussian quadrature program is not truly fair for two main reasons, namely:

- Firstly, you're comparing a microcode routine versus an RPN, user code routine, which is bound to execute significantly slower. The proper criterium would be to compare the actual number of function evaluations each routine needs to achieve the result.

  You specified 20 subintervals with my routine, which translates to 60 function evaluations. The HP-15C's built-in Romberg method, working at FIX 5, does need exactly 63 function evaluations for this particular f(x), and the FIX 8 result probably needs 511 function evaluations.

- Secondly my routine was written with the principal criterium being that it would fit in the least possible amount of RAM and use a minimum of resources, while providing the fastest possible running time and accuracy. It just accepts a user-specified number of subintervals, and merrily goes computing each subinterval with a 5th-degree fitting, 3-point Gaussian quadrature, totalling each partial result and outputting the grand total at the end.

  On the other hand, the HP-15C's Romberg method uses a much more convoluted approach to cater for its low degree, continually halving the subintervals, taking note of the different results obtained by using, say, 8 and 16 subintervals, then performing an extrapolation (series acceleration) to predict the final iteration without actually carrying it out. That way it starts with 1, 2, 4, 8 function evaluations, continually computing the relative errors, predicting the one for the following doubling process, then stopping when it's deemed below the threshold. That's why INTEG takes so many resources to run (23 registers, IIRC, versus 5 for my routine).

Comparing this much more elaborated, microcoded process against my simpler-by-design RPN routine is thus unfair to the Gaussian approach. It's actually very easy to do as the HP-15C does, i,.e.: start with one subinterval, then halving it continuously, then using series acceleration to predict the final integral while keeping also a prediction of the diminishing error. If that were to be done, you'd see the Gaussian approach easily outperforming the HP-15C built-in routine.

Best regards from V.

## thanks

*Message #23 Posted by **bill platt** on 19 June 2006, 2:10 p.m.,*
*in response to message #1 by Gerson W. Barbosa*

Hi Gerson,

Thanks for posting your routine.

In ship design, we all grew up learning to use Simpson's rule to calculate integrals (waterplane areas, volumes, inertias etc) and so it is fun to see your implementation. (of course the Simpson's rule I am talking about can be done perfectly well on paper using 1,4,2,4,1 factors and a 1/3 overall factor. I programmed my 11c to do that when I was 19).

Interestingly, nobody ever mentioned trying to use a Gaussian approach.

Now that we have digital computers, we let them do the numerical integration for us, and we generally have a surface model over which we are cutting a plane or a volume to integrate. Currently, the most widely used program actually uses the *trapezoidal* rule for this! My favorite program for initial design uses the Simpson's rule, and therefore requires far fewer integration "stations."

I'll have to ask the program author, next time I see him, what he would think of the Gaussian approach. There may be a hitch.

The other great thing about your post, Gerson, is that it elicited a great response from Valentin! (thanks, V. :-)

*Edited: 20 June 2006, 9:19 a.m. after one or more responses were posted*

## Gaussian vs. Simpson's 1/3

*Message #24 Posted by **Valentin Albillo** on 20 June 2006, 9:52 a.m.,*
*in response to message #23 by bill platt*

Hi, Bill:

Bill posted:

*"The other great thing about your post, Gerson, is that it elicited a great response from Valentin! (thanks, V. :-)"*

> You're welcome, thanks should go to you for your continued interest.

As for comparison between Simpson's 1/3 and Gaussian, have a look at this URL, which explains it all in a very simple and didactic way, does include a worked out example and a neat comparison table, then ends with this comment (the underlining is mine):

> *"While this example is quite simple, the following table of values obtained for N ranging from 2 to 10 indicates how accurate the estimate of the integral is for only a few function evaluations. [...] The Gauss-Legendre result is correct to almost twice the number of digits as compared to the Simpson's rule result for the same number of function evaluations."*

As for why anyone would prefer or choose to implement Simpson's rather than Gaussian, it's probably due to the fact that Simpson's rule is more easily explained (parabolic arcs), its coefficients and weights are small integers which require next to no RAM to store, and it's actually trivial to implement.

On the other hand, Gaussian quadrature, while being equally simple to implement, does require having the x-ordinates and weights precomputed to the desired number of decimal places (or else compute them on the fly), which wasn't that easy to do in a calculator 20 years ago, and further, they do take some amount of RAM to store, which was pretty scarce in calculators back then.

So, most textbooks of the time simply mentioned Gaussian quadrature, if at all, then went forth with the (then) much simpler Simpson's rule. Educational inertia did the rest, but this wasn't actually true then, as my short HP-11C routine above makes pretty clear, and it isn't true now, when, for instance, my 17-line routine can do a 8192-point (computed on the fly) Gaussian quadrature over 65536 subintervals or more and get results accurate to thousands of decimal places in mere seconds on any run-of-the-mill, inexpensive PC.

As for calculators, any decent model, even vintage ones, can do a 16-point, say, Gaussian quadrature which would provide impressive accuracy using very few subintervals, thus very quickly. In Gerson's example, we've seen this gives up to 48 correct digits without the need to split the (0,2) interval at all.

Best regards from V.

## Re: Gaussian vs. Simpson's 1/3

*Message #25 Posted by **Kiyoshi Akima** on 20 June 2006, 6:32 p.m.,*
*in response to message #24 by Valentin Albillo*

Also, back in those days, a lot of integration was being done on stuff out of tables. Gaussian quadrature works fine if you can control the X-values and compute the function anywhere. If you're limited to looking up and/or computing function values at a fixed set of (usually equally-spaced) points, your choices become limited. If you have an even number of intervals, you can use Simpson. If you don't, you may have to drop back to the Trapezoidal. And if the points aren't equally spaced...

## Re: Gaussian vs. Simpson's 1/3

*Message #26 Posted by **bill platt** on 20 June 2006, 9:51 p.m.,*

*in response to message #25 by Kiyoshi Akima*

(and if the points ar eequally spaced)...then there are the Tchebycheff's rules, which use unequal spaces.

The convenience of equal stations was of great importance in computing volumes and other integrals in ship design, as it made it possible to have a set number of fixed "stations" from which to measure sectional areas. However, the sticky wicket was always thje problem of what to do at the "end" of the integration range--as the waterplane or waterline etc would not conviniently land on a fixed station in all attitudes, and so either new stations, or interpolated stations, would be used for developing the appropriate integration values.

[ Return to Index | Top of Index ]

GTO Go back to the main exhibit hall