My Favorite HP Prime Functions

Namir Shammas

Introduction

HP has chosen to launch the new HP Prime using much publicity and fanfare. This type of promotion seems to indicate that HP is committed to producing calculators that once again will lead the pack of calculator manufacturers. In fact, beta versions of the emulator for the HP Prime were released to the interested public. HP has benefited from this strategy by collecting valuable suggestions and bug reports. All of this feedback aims at raising the users' anticipation for the new product and reducing bugs in the new graphing calculator.

The HP Prime graphing calculator is essentially a handheld mathematical beast! The product packs a lot of punch. The documentation for the HP Prime runs over 600 pages, discussing a whole myriad of features, apps, and built-in functions.

The HP Prime operates in two major modes, namely the Home mode and CAS mode. When you press the **Menu** button (located under the **Num** button) you get the **Math** menu (that supports Home-mode functions), the **CAS** menu, and other menus. I will discuss groups of functions that are in both the Home and CAS modes, which I found to be interesting. I will give very brief examples for these functions. My aim is to give you a taste and whet your appetite if you have not been already exposed to the HP Prime. The list of functions that I cover is purely subjective and somewhat arbitrary, influenced by functions that I have used in the past.

The Probability Functions

Figure 1 shows the probability functions that are part of the **Math** menu. Note the last three submenus in the **Probability** menu—the **Density**, **Cumulative**, and **Inverse**. All three submenus yield similar options to work with the Normal, Student-t, Chi-Square, Snedecor's F, Binomial, and Poisson distributions.

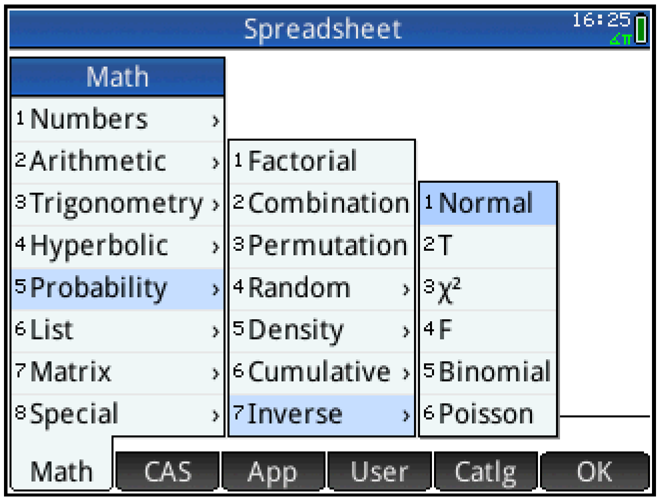
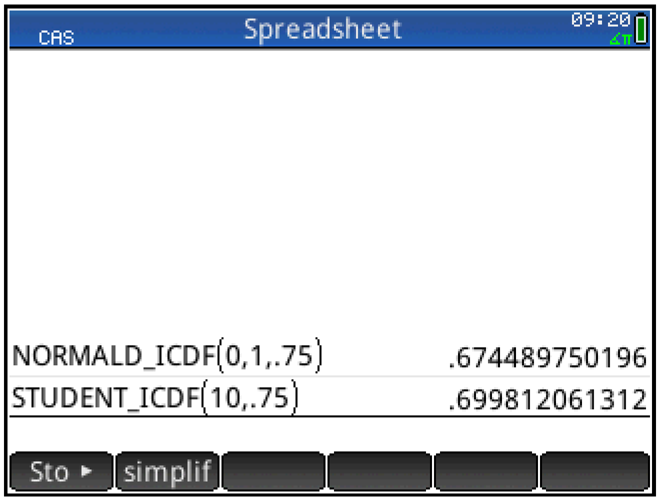


Figure 1 – The Probability Functions.

Of particular interest to all who perform statistical hypothesis calculations are the Normal, Student-t, Chi-Square, and Snedecor's F inverse cumulative distribution functions (which I will abbreviate using ICDF). These functions replace legacy tables of ICDF values found in just about every statistical book. Using the ICDFs allows you to wave goodbye to the legacy ICDF tables. Figure 2 shows examples for the Normal and Student-t ICDFs. The figure shows the Normal ICDF for a distribution with a mean of 0, standard deviation of 1, and probability of 0.75. In addition, the figure shows the Student-t ICDF for a distribution with 10 degrees of freedom and 0.75 probability.



```
NORMALD_ICDF(0,1,.75)      .674489750196
STUDENT_ICDF(10,.75)       .699812061312
```

Figure 2 – Using the Normal and Student ICDFs.

The Matrix Functions

The **Math** menu offers you a plethora of matrix functions that seasoned users of programmable calculators would have only dreamt of having some 30 or 40 years ago. Figure 3 shows the **Matrix** menu with it submenus, showing the options for the **Create** submenu. I found the **Random** and **Vandermonde** functions to be interesting. The latter function creates a matrix **X** that can be used with a vector **y** to calculate the coefficients of a polynomial. The number of columns in the Vandermode matrix equals the order of the resulting polynomial plus one.

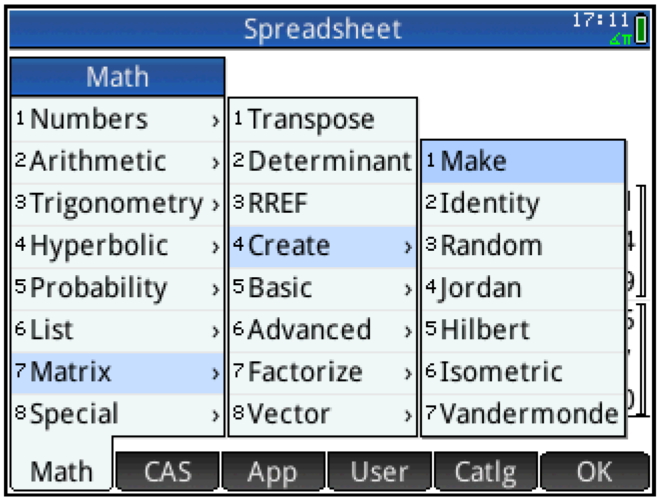
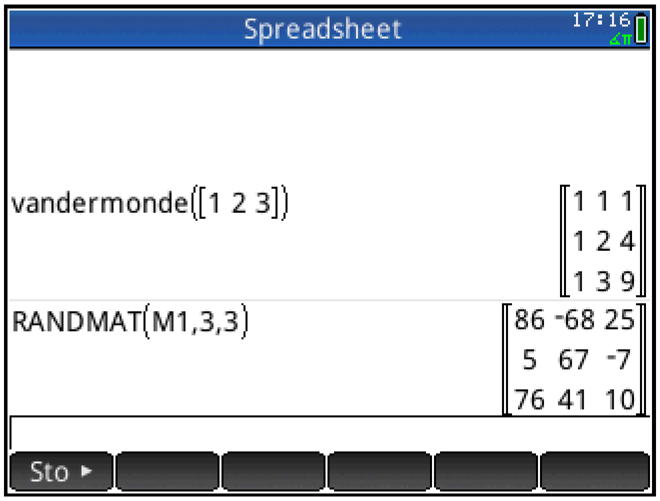


Figure 3 – The Create submenu Functions.

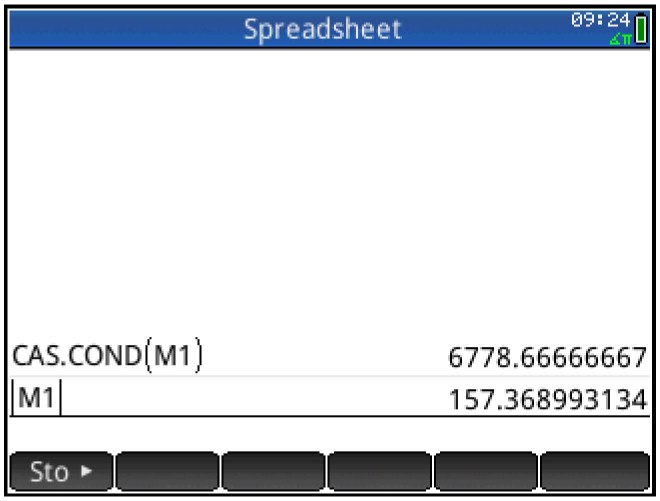
Figure 4 shows sample calls to functions **vandermonde** and **RANDMAT**. The call to function **vandermonde** supplies an argument that is an array of three values of x. The function returns a Vandermonde matrix where the first column has 1s, the second column has values that match the input array, and the third column has values that are the squares of the input array. The call to function **RANDMAT** specifies that the output matrix is **M1** having a 3 by 3 matrix of random integers with values between -99 and 99.



```
vandermonde([1 2 3])          [[1 1 1]
                               [1 2 4]
                               [1 3 9]]
RANDMAT(M1,3,3)          [[86 -68 25]
                          [ 5  67 -7]
                          [76  41 10]]
```

Figure 4 – Sample calls to function vandermonde and RANDMAT.

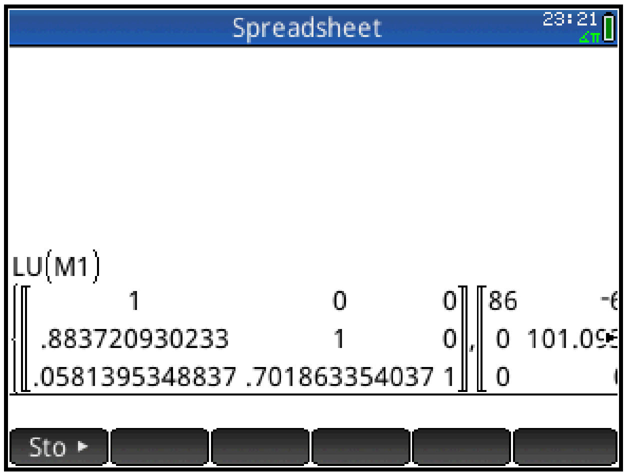
The **Matrix** menu has the **Basic** submenu that calculates various types of norms and matrix properties. I find the **Norm** and **Condition** functions to be very interesting. The **Norm** function is usually used to calculate a summary of errors in iterative calculations (like optimization problems and the iterative solutions of very large systems of linear equations). The **Condition** function (which is a bit complex to implement from scratch) returns the condition of a matrix. This value indicates how easy it is for the matrix to yield solutions of linear equations. The lower the condition number is, the easier it is to solve a system of linear equations with the matrix. Figure 5 shows a call to the **COND** function using the matrix **M1**. The figure also shows the calculation of the norm of matrix **M1**.



```
CAS.COND(M1)          6778.66666667
|M1|                  157.368993134
```

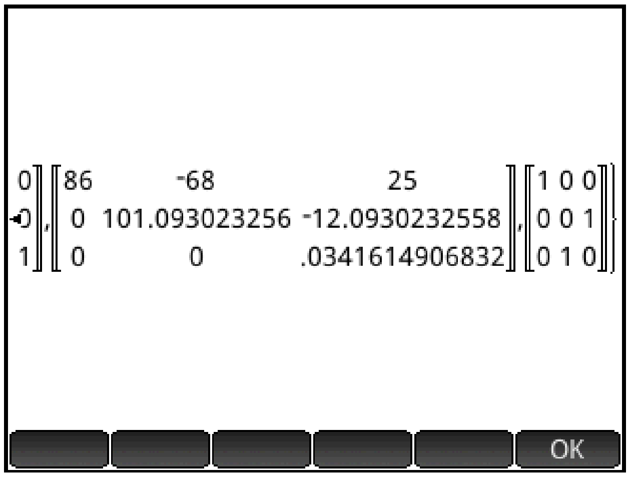
Figure 5 – Using the COND and norm functions.

The **Matrix** menu has the **Factorize** submenu that calculates various types of factorized matrices. I find the **LSQ**, **LU**, **QR**, and **SVD** to be versatile and powerful functions. Regarding the **LSQ** function, this function returns the least-squares regression coefficients of an independent-variable matrix **X** and a dependent-variable vector **y**. The January 2013 issue of *HP Solve* presented a series of four articles, aimed at the HP39gII, that show you how powerful and versatile function **LSQ** is. The functions **LU**, **QR**, and **SVD** perform various kinds of matrix factorizations used in solving systems of linear equations. The functions **QR** and **SVD** handle difficult factorization cases. Here is an example for using function **LU** with matrix **M1** that I created earlier. The **LU** function factorizes the matrix argument into a lower matrix L, upper matrix U, and index matrix P. Typing **LU(M1)** at the command line yields Figures 6 and 7 which show the three output matrices.



```
LU(M1)
[[[1               0              0]  [86  -6
  [.883720930233   1              0], [0   101.09
  [.0581395348837  .701863354037  1]] [0
```

Figure 6 – Sample call to function LU showing output matrix L.



```
0]] [[86   -68            25            ]] [[1 0 0]]
0]],  [0   101.093023256  -12.0930232558], [0 0 1]
1]]   [0   0              .0341614906832]]  [0 1 0]]
```

Figure 7 – Sample call to function LU showing output matrices U and P.

You can combine the **vandermonde** and **LSQ** functions in the following short program to calculate the coefficient of an interpolating polynomial. The order of that polynomial is equal to the number of data points minus one. The program **VandPoly** is:

```
EXPORT VandPoly(x,y)
BEGIN
  LOCAL xm;
  xm:=CAS.vandermonde(x);
  RETURN LSQ(xm,y);
END;
```

As an example, I can use the program **VandPoly** to calculate the coefficients of the quadratic polynomial that passes through the three points (1, 6), (2, 11), and (3, 18). By executing the following command:

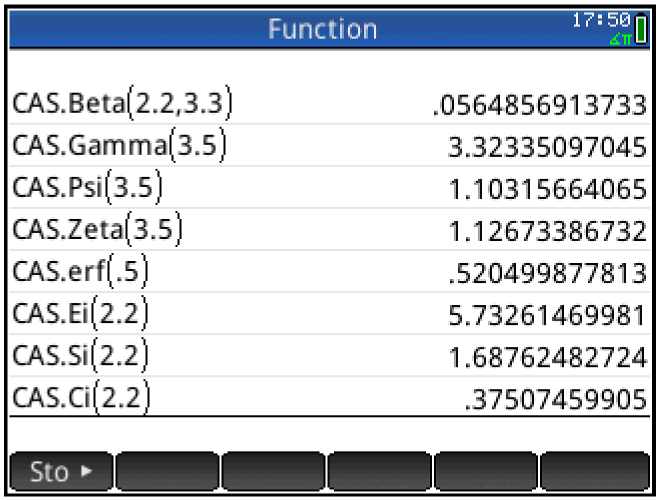
```
VandPoly([1 2 3][6 11 18])
```

I get the array [3 2 1] which means that the interpolating quadratic polynomial is:

$y = 3 + 2x + x^2$

The Special Functions

The **Math** menu has the **Special** submenu which offers special functions. These functions are the Beta, Gamma, Psi, Zeta, error function, complimentary error function, exponential integral, sine integral, and cosine integral. The Gamma and error function are perhaps the most popular functions in the list. Figure 8 shows sample calls to most of the functions in the **Special** submenu.



```
CAS.Beta(2.2,3.3)          .0564856913733
CAS.Gamma(3.5)             3.32335097045
CAS.Psi(3.5)               1.10315664065
CAS.Zeta(3.5)              1.12673386732
CAS.erf(.5)                .520499877813
CAS.Ei(2.2)                5.73261469981
CAS.Si(2.2)                1.68762482724
CAS.Ci(2.2)                .37507459905
```

Figure 8 – Sample calls to functions in the Special submenu.

Working with the Solve Functions

The **CAS** menu has several submenus, including the **Solve** submenu. Figure 9 shows the **Solve** submenu. This submenu offers functions that solve for the roots of polynomials, general equations, differential equations, and systems of linear equations.

Figure 10 shows sample calls to various root-seeking functions that calculate the roots of polynomials and non-polynomials. The first few function calls solve for the real and complex roots of polynomials. The last two calls invoke function **nSolve** to numerically calculate the two different roots for the equation exp(x)-3*x^2=0 at near the initial guesses of 5 and -1.

Working with the Polynomial Functions

Figure 11 shows the options for the **Special** submenu. These options support various kinds of particular polynomials. Among them are the Hermite, Lagrange, Laguerre, Legendre, and Chebyshev orthogonal polynomials. These polynomials represent the solutions for various classes of ordinary differential equations. They also play a key role in the various types of Gaussian quadrature. By solving for the roots of these orthogonal polynomials you can calculate the nodes (also known as abscissa points) and their

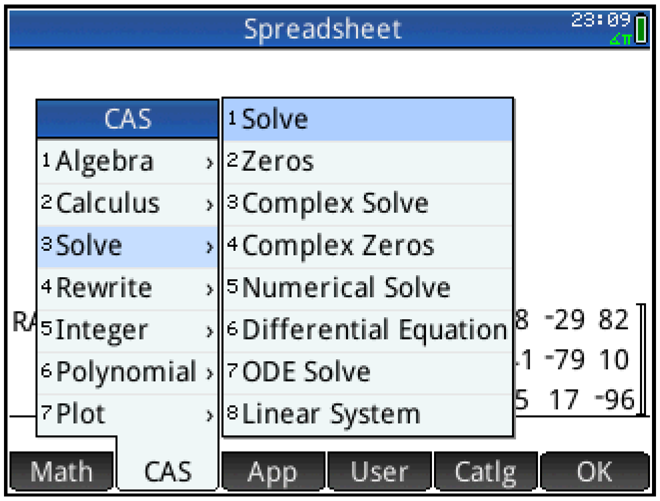
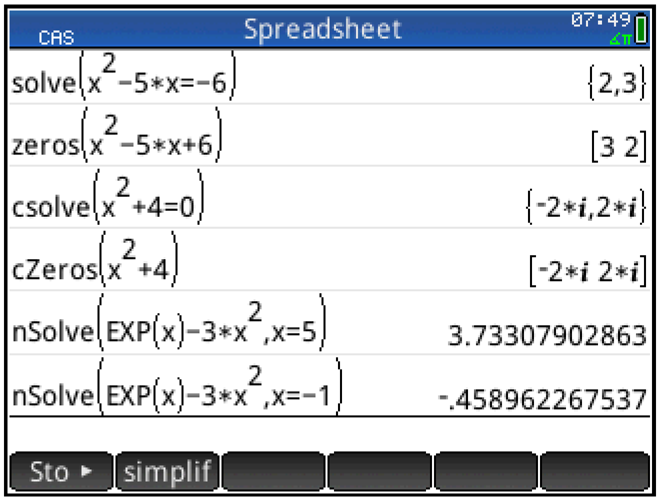


Figure 9 – The Solve submenu.



```
solve(x^2-5*x=-6)                    {2,3}
zeros(x^2-5*x+6)                     [3 2]
csolve(x^2+4=0)                 {-2*i,2*i}
cZeros(x^2+4)                   [-2*i 2*i]
nSolve(EXP(x)-3*x^2,x=5)     3.73307902863
nSolve(EXP(x)-3*x^2,x=-1)    -.458962267537
```

Figure 10 – Calculating the roots for various equations.

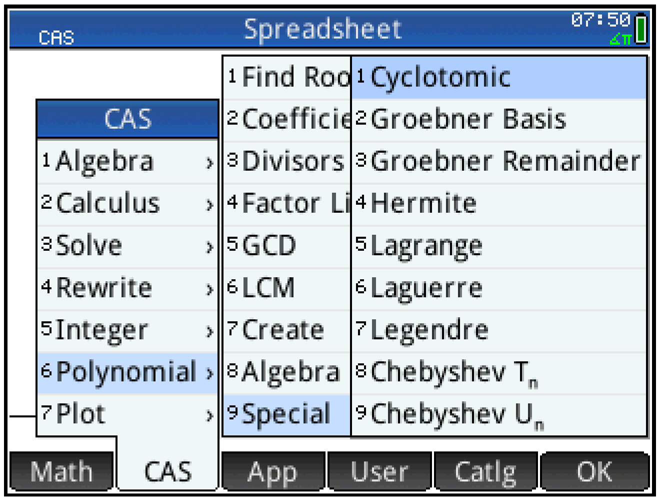
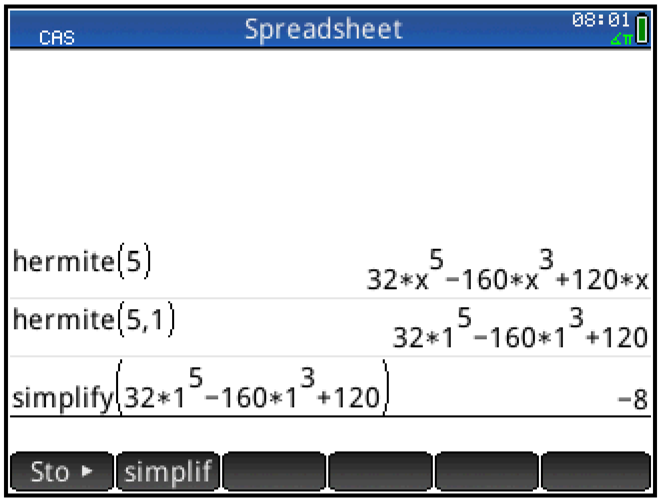


Figure 11 – The special polynomials.

associated weights used in evaluating the Gaussian quadrature. The HP Prime allows you to use these functions to obtain the polynomial equation for the order you specify. Alternatively, you can include a value for the polynomial and have the function return a value. The HP Prime is the first calculator that supports the above orthogonal polynomials.

Figure 12 shows sample calls to the Hermite polynomial. The first call requests that the calculator returns the polynomial expression for the 5th order Hermite polynomial. The second call includes the value for x and tells the calculator to return 5th order Hermite polynomial evaluated at x=1. The last command tells the calculator to simplify the previous result and give us a single number.

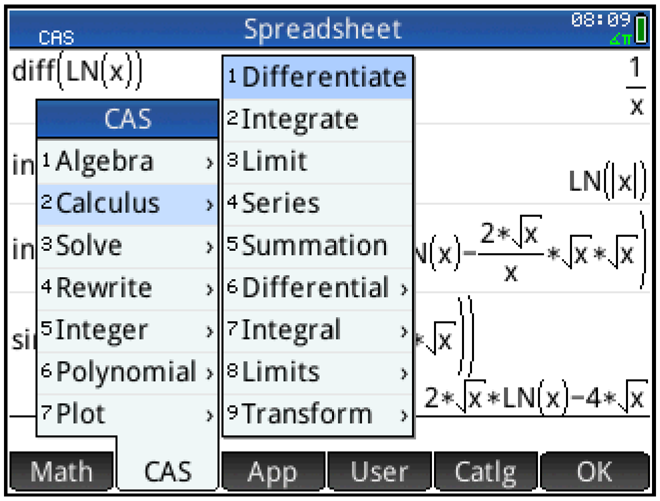


```
hermite(5)                          32*x^5-160*x^3+120*x
hermite(5,1)                          32*1^5-160*1^3+120
simplify(32*1^5-160*1^3+120)                          -8
```

Figure 12 – Sample calls to the Hermite function.

Symbolic Differentiation and Integration Functions

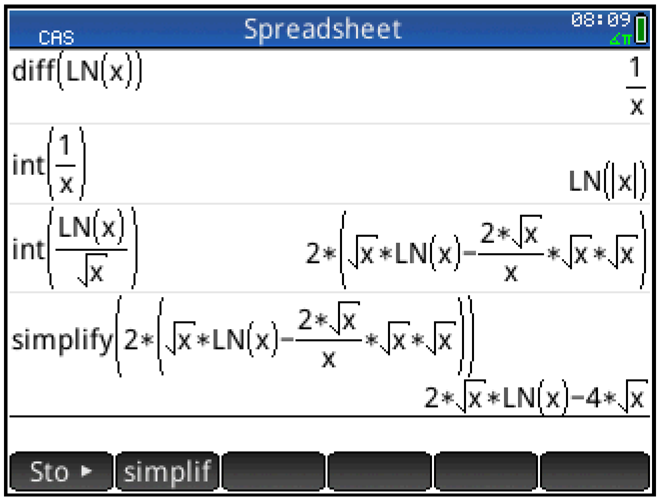
The **CAS** menu has the **Calculus** menu, shown in Figure 13, which shows several submenus for various calculation operations. The first two options are among the ones that I find very interesting, allowing the CAS system to do best what we expect it to.



```
diff(LN(x))                                          1/x
                                                 LN(|x|)
                                   2*√x*LN(x)-4*√x
```

Figure 13 – The Calculus menu and its submenus.

Figure 14 shows examples for symbolic differentiation and integration. The call to the **diff** function returns 1/x as the derivative of the natural logarithm. The call to the integral function **int** integrates 1/x to yield the natural logarithm. The second call to function **int** integrates the function LN(x)/√x to yield 2(√x*LN(X)-2*√x/x*√x*√x). The figure shows that I used the **simplify** menu to simplify the last result into 2*(√x*LN(X)-4*√x).



$\mathrm{diff}(\mathrm{LN}(x))$ $\quad \frac{1}{x}$

$\mathrm{int}\left(\frac{1}{x}\right)$ $\quad \mathrm{LN}(|x|)$

$\mathrm{int}\left(\frac{\mathrm{LN}(x)}{\sqrt{x}}\right)$ $\quad 2*\left(\sqrt{x}*\mathrm{LN}(x)-\frac{2*\sqrt{x}}{x}*\sqrt{x}*\sqrt{x}\right)$

$\mathrm{simplify}\left(2*\left(\sqrt{x}*\mathrm{LN}(x)-\frac{2*\sqrt{x}}{x}*\sqrt{x}*\sqrt{x}\right)\right)$

$2*\sqrt{x}*\mathrm{LN}(x)-4*\sqrt{x}$

Figure 14 – Performing symbolic differentiation and integration.

Using the HP Prime Emulator as an SDK

The HP Prime emulator is a valuable tool in developing programs for the HP Prime calculator. You can use your favorite text editor on your PC to create the source code programs as text files. These text files also serve as backup. When you create a new program in the emulator, you can delete the skeleton code that the emulator inserts and copy the source code from your text editor into the emulator using the **Edit | Paste** menu option. Once you make sure that the program in the emulator is bug free and works the way you want it, you are ready to transfer your program to the physical calculator. You connect the physical HP Prime calculator to your PC via the supplied USB cable. Clicking the **Send** menu on the emulator sends the currently selected program effortlessly to the physical HP Prime. You are now ready to run your new program on the physical calculator!

Observations and Conclusions

The HP Prime packs some serious punch by supporting various features, operations, apps, and functions. This article gave you a small taste of the functions built-in the HP Prime. This new graphing machine brings with it new graphing features and new functions not available on any other hand-held calculator.

References

1. Namir Shammas, "HP39gII Regression: Part I", HP Solve, Issue 30, January 2013.
2. Namir Shammas, "HP39gII Regression: Part II", HP Solve, Issue 30, January 2013.
3. Namir Shammas, "HP39gII Regression: Part III", HP Solve, Issue 30, January 2013.
4. Namir Shammas, "HP39gII Regression: Part IV", HP Solve, Issue 30, January 2013.
5. Abramowitz and Stegun, "Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables", 1965, Dover Publishing.
6. Olver, Lozier, Boisvert, andsClark, "NIST Handbook of Mathematical Functions, 2010, Cambridge University Press.

7. Press, Teukolsky, Vetterling, and Flannery, "Numerical Recipes 3rd Edition: The Art of Scientific Computing", 2007, Cambridge University Press.

About the Author

Namir Shammas is a native of Baghdad, Iraq. He resides in Richmond, Virginia, USA. Namir graduated with a degree in Chemical Engineering from the University of Baghdad. He also received a master degree in Chemical Engineering from the University of Michigan, Ann Arbor. He worked for a few years in the field of water treatment before focusing for 17 years on writing programming books and articles. Later he worked in corporate technical documentation. He is a big fan of HP calculators and collects many vintage models. His hobbies also include traveling, music, movies (especially French movies), chemistry, cosmology, Jungian psychology, mythology, statistics, and math. As a former PPC and CHHU member, Namir enjoys attending the HHC conferences. Email me at: nshammas@aol.com