Adapting the Maple Computer Algebra System to the Mathematics Curriculum*

J. S. Devitt

The University of Saskatchewan

Introduction

This year has seen considerable excitement in the mathematical community over the increased availability of symbolic algebra software for use in the classroom. It is not that the software has just emerged. In fact, such software has been under development since the early 1960's [7,6]. Rather, it is a combination of issues including the evolution of the software and hardware that combine to make sufficiently powerful versions of the software available to individual students and instructors.

Numerous attempts to include computation in the curriculum have been made over the past decade using a variety of software and hardware platforms. However, the ready availability of computational support for algebra is a key step towards providing an adequate computational environment for doing college and research level mathematics. It is not intended as a replacement for the computational mainstays of graphics or numerics any more than algebra is intended as a replacement for numerical computation or visual representation. The point is that algebraic computation plays a crucial role in modelling and in analyzing and setting up problems. In describing our visual and numerical observations we often rely heavily on algebra to state and analyze the underlying properties of the model? With symbolic algebra we provide the computational support necessary to derive and manipulate this underlying model.

Much of the past development effort for computer algebra systems has gone into extending them to carry out large and difficult algebraic computations. Considerable progress is being made in this direction but there is more to doing mathematics than completing large computations. As these systems mature it is possible to place some emphasis on using them to do mathematics. This places different demands on them.

For example, we may want to transform an indefinite integral using integration by parts rather than evaluating it. Computer algebra systems must allow us to manipulate and observe algebraic entities from a variety of points of view.

This article raises some of the issues about the style of use of computer algebra systems through a variety of examples based on the use of Maple^{TM} to solve particular problems arising from first year calculus. [1] These examples run using Maple version 4.2 on, for example, a standard Macintosh Plus with one megabyte of memory. [2] They emphasize how the Maple system can and has been adapted to the needs which arise in talking about and teaching mathematics. The examples do not come close to describing the full computational power of Maple. Our aim is to illustrate a philosophy or style of use appropriate for such tools.

Daily use in the classroom

Ideally we should be able to use a symbolic algebra system in an interactive session much like we would the blackboard and chalk. A live symbolic algebra session projected by means of an overhead projector can be the focus of every lecture. At this point the black board may still be used to sketch rough diagrams motivating concepts and to state theorems. However, almost all computation can be done by machine.

- Students observe first hand how to carry out certain basic tasks within the system.
- The instructor role model supports the overall goals of applying the available tools to the problem at hand. We don't imply that "Computers aren't much use for what I do!" by avoiding the machine at every possible turn.
- The focus of the lectures is on the structure of problem solving rather than on the details of every computation.

[1] Maple is a trademark of the University of Waterloo.

[2] Maple: Symbolic computation for the Macintosh is now available from Brooks/Cole Publishing Co., Monterey, CA. Macintosh is a registered trademark of Apple Computer Inc.

Through the use of online documents containing a mixture of explanations and system commands, we can extend this classroom experience to the laboratory.

When is a solution a solution?

A question which has been forcefully raised by the "Four Colour Theorem" [8] is "What is the nature of a mathematical solution?".

Symbolic algebra systems cause a real dilemma. Consider the solutions to the following problems proposed in the American Mathematical Monthly.

Compare the following sum and product [9]

```
#--> Sum(
#--> (-1)^(k-1)*binomial(N,k)*k/(1+(k-1)*a),
#--> #--> k=1..N)
#--> #--> =
#--> #--> Product( (k+1) / (k+1/a), k=1..N-1);

   N
 -----        (k - 1)
  \       (-1)        binomial(N, k) k
   )    ------------------------------
  /              1 + (k - 1) a
 -----
 k = 1

          N - 1
        --------'
       ' | |    k + 1
    =    | |  ---------
         | |   k + 1/a
         | |
        k = 1
```

Here the Sum and Product deliberately returned *unevaluated* though of course the algebra system may be unable to complete a computation and so would return unevaluated of its own accord.

We next force Maple to try evaluating the above expression using a routine which effectively replaces "Sum" by "sum", etc. [3]

```
#--> Eval(");

          1 + a              1 + a
 N GAMMA(-------)  GAMMA(------- - 1/a + N - 1)
            a                  a
----------------------------------------------
         1 + a                 1 + a
  GAMMA(------- - 1/a) GAMMA(------- + N - 1)
           a                     a
      GAMMA(N + 1) GAMMA(1 + 1/a)
   = ----------------------------
             GAMMA(N + 1/a)
```

[3] There is a convention that uppercase equivalents of lowercase commands are inert. `Eval()` is a function in the student package.

We appear to be successful and the answer can be further simplified further as:

```
#--> simplify(");

      1 + a
 GAMMA(-------) GAMMA(N + 1)
        a
------------------------------
          1 + N a
    GAMMA(---------)
             a

          1 + a
    GAMMA(-------) GAMMA(N + 1)
            a
  = ------------------------------
              1 + N a
        GAMMA(---------)
                 a
```

The name """ stands for "the previous expression".
Or consider this ordinary differential equations problem.[10]

```
#--> ODE := y3 = y1*(3*y2^ 2 - y3*y1);

                                          2
              ODE := y3 = y1 (3 y2 - y3 y1)

#--> y1 := diff(y(x),x);

                               d
                      y1 := ---- y(x)
                              dx

#--> y2 := diff(y(x),x,x);

                                2
                               d
                      y2 := ----- y(x)
                                2
                              dx

#--> y3 := diff(y(x),x,x,x);

                                3
                               d
                      y3 := ----- y(x)
                                3
                              dx

#--> #--> dsolve(ODE,y(x));

y(x)

         2       2                 2       2     2    1/2
  = ((x   exp(C) + 2 x exp(C) C1 + C1 exp(C) - 1)

             2
     / exp(C)

     - C1
```

```
       ln(2

         exp(C)

              2         2               2          2        2
            (x  exp(C)  + 2 x exp(C)  C1 + C1  exp(C)  - 1)

             ^ (1/2)

                    2                  2
        + 2 x exp(C)  + 2 exp(C)  C1)

        1 / exp(C))

         1/2
     (-1)     exp(C)

     + ln(2

        exp(C)

             2     2            2          2       2
          (x exp(C)  + 2 x exp(C) C1 + C1 exp(C)  - 1)

            ^(1/2)

                   2                   2
       + 2 x exp(C)  + 2 exp(C)  C1)

           1/2
       (-1)    C1

    + C2
```

Though the final answer is not as elegant as we might like, it can be simplified.

```
#--> subs( exp(C)=K,");

y(x)

           2 2       2            2 2   1/2
         (x K + 2 x K C1 + C1 K - 1)
    = (-----------------------------------
                             2
                             K

      - C1

              2  2         2        2  2     1/2          2
      ln(2 K (x K + 2 x K   C1 + C1   K   - 1)     + 2 x K

             2
         + 2 K C1)

         1 / K)

         1/2
      (-1)     K

                2  2          2       2  2      1/2         2
     + ln(2 K (x K + 2 x K C1 + C1   K    - 1)  2+ 2 x K

            2
        + 2 K C1)
```

```
         1/2
     (-1)    C1

   + C2
```

In both these examples, we observe that:

- The user has control over what evaluation is attempted and when it takes place.
- The system supports a high level of abstraction providing a near 1-1 correspondence with the tasks at hand.
- The system is fairly powerful.
- We will need to develop verification techniques if we are to find these answers acceptable.

The verifications refered to above will benefit from machine computation as well. For example, we could substitute the solution to the differential equation back into the original equation using the command `subs(",ODE);`.

Some sample symbolic computations

We begin by reviewing some of the basic capabilities of Maple. All the examples you see here run on the basic Mac Plus with only one megabyte of memory. The actual Maple commands used are indicated here by use of the prompt `#-->`.

The overall capabilities of a symbolic algebra system can be quite impressive. First noticed may be the freedom from concern about machine specific restrictions such as limits on the size of integers and refusals to represent rationals exactly.

```
#--> 2/3;

                                2/3

#--> 100!;

933262154439441526816992388562667004907159682643816214685929\
638952175999932299156089414639761565182862536979208272237582\
51185210916864000000000000000000000000

#--> evalf(Pi,100);

.3141592653589793238462643383279502884197169399375105820974\
944592307816406286208998628034825342117068
   * 10 ^ 1
```

Essential operations such as integer greatest common divisors and factoring are made available interactively.

```
#--> 3^ 500;

36360291795869936842385267079543319118023385026001623040346\
03583258060019158389548419850826297938878330817970253440385\
57528559315170130661429924309165620257800217712478476434501\
25342836565813209972590371590152578728008385990139795377610\
001

#--> igcd(",5^ 300);

                                 1
```

Our computational experience might have convinced us that we should never leave a variable unassigned, but here the variable may well be the data.

```
#--> s1 := Sum(i^ 20,i=1..n);

                                    n
                                  -----
                                   \     20
                          s1 :=     )    i
                                   /
                                  -----
                                  i = 1
#--> s2 := Sum(1/i^ 3,i=1..infinity);

                                infinity
                                  -----
                                   \        1
                          s2 :=     )      ----
                                   /        3
                                  -----    i
                                  i = 1
```

Unevaluated expressions like these can be evaluated. exactly by Maple.[4]

```
#--> Eval(s1);

             21               20                 19
1/21 (n + 1)   - 1/2 (n + 1)   + 5/3 (n + 1)

                    17    1292         15                13
   - 19/2 (n + 1)    + ------ (n + 1)   - 323 (n + 1)
                          21

       41990         11     223193          9                7
   + ------- (n + 1)    - -------- (n + 1) + 6460 (n + 1)
       33                    63

     68723          5    219335          3    174611
   - ------- (n + 1) + -------- (n + 1) - -------- n
     10                   63                  330

     174611
   - --------
      330

#--> Eval(s2);

                           Zeta(3)
```

The Maple system generally assumes it is working with polynomials having exact rational number coefficents. However, we can choose to work with algebraic extensions of the rationals.

```
#--> f := expand( (x^ 2 - (3*alpha-1)*x + alpha^ 2)
#--> * (x-2*alpha)^ 2 );

      4     3               2     2            3     3
f := x - 7 x alpha + 17 x alpha - 16 alpha x + x

              2             2       4
   - 4 alpha x + 4 x alpha + 4 alpha
```

[4] the Eval() command is a command from the student package which is part of the basic maple system and is used to convert from unevaluated forms of expression to ones which evaluate.

```
#--> g := expand( (x^ 2+x+1)^ 2 * (x-2*alpha)^ 2 );

      6     5              4     2     5     4
g := x - 4 x alpha + 4 x alpha + 2 x - 8 x alpha

        3     2     4      3              2      2      3
   + 8 x alpha + 3 x - 12 x alpha + 12 x alpha + 2 x

           2              2   2                     2
   - 8 alpha x + 8 x alpha + x - 4 alpha x + 4 alpha

#--> alpha := RootOf(x^ 5+x^ 3+1,x);

                                         5   3
               alpha := RootOf(_Z + _Z + 1)

#--> evala(Gcd(f,g));

    2             5   3                        5   3   2
   x - 4 RootOf(_Z + _Z + 1) x + 4 RootOf(_Z + _Z + 1)
```

The library of about a thousand routines organized into man subdirectories. subdirectories. Most of these are automatically loaded when needed. However, some mechanism must be provided for focussing on a specific subject area such as linear algebra, first year calculus, or number theory. An environment for linear algebra can be quickly constructed by the `with()` command which defines (but does not load) a library of routines on this specific topic.

```
#--> with(linalg);

Warning: new definition for trace
[jacobian, band, smith, add, vectdim, trace, gausselim,

   orthog, laplacian, transpose, cond, rowspace, leastsqrs,

   scalarmul, adj, genmatrix, hadamard, dotprod, mulcol,

   swaprow, vandermonde, submatrix, det, swapcol,

    singularvals, bezout, definite, hilbert, range, mdet,

   kernel, linsolve, indexfunc, diverge, hessian, addrow,

   sylvester, multiply, adjoint, nullspace, mulrow,

   inverse, rank, subvector, rowdim, toeplitz, angle,

   ismith, eigenvals, colspace, addcol, crossprod, grad,

   norm, coldim, curl, permanent]
```

Once defined the routines will be automatically loaded when needed. These generally apply to matrices with symbolic entries.

```
#--> jacobian([1/sin(x+y)],[x,y]);

array ( 1 .. 1, 1 .. 2,

                 cos(x + y)       cos(x + y)
            [- -------------, - -------------]
                          2                2
                sin(x + y)       sin(x + y)

)
```

Much of the work in designing computer algebra systems has gone into developing sophisticated and powerful methods to attack large computational problems. Indefinite integration has received considerable attention. The Risch integration algorithm is used for many classes of problems.

```
#--> f1 := x / (exp(x) + 1);

                              x
                  f1 := ------------
                         exp(x) + 1

#--> int(f1,x);

                     /
                    |       x
                    |  ------------ dx
                    |   exp(x) + 1
                   /
```

Though it is not obvious in this example, Maple has actually *proved* that the above integral cannot be expressed in terms of elementary functions.[5] Again we are faced with the question "What constitutes a proof?". Our ability to comprehend and verify solutions becomes essential. The following slightly different integral can be expressed in terms of elementary functions;

```
#--> f2 := 1 / (exp(x) + 1);

                              1
                  f2 := ------------
                         exp(x) + 1

#--> int(f2,x);

              - ln(exp(x) + 1) + ln(exp(x))
```

as can the following integral.

```
#--> num := x*(x+1) * ( (x^ 2*exp(x^ 2)^ 2 - log(x+1)^ 2)^ 2
#--> + 2*x*exp(x^ 2)^ 3 * (x - (2*x^ 3+2*x^ 2+x+1)*log(x+1)) ):
#--> den := ( (x+1)*log(x+1)^ 2 - (x^ 3+x^ 2)*exp(x^ 2)^ 2 )^ 2:
#--> f3 := num/den;

f3 :=

   x (x + 1)

          2     2 2            2 2
      ((x exp(x )  - ln(x + 1) )

         + 2

                  2 3          3     2
          x exp(x ) (x - (2 x + 2 x + x + 1) ln(x + 1))

          )

                          2     3   2        2 2 2
      / ((x + 1) ln(x + 1)  - (x + x ) exp(x ) )
```

[5] An unevaluated return in Maple generally means that Maple has been unable to solve the problem. This integral happens to fall into the class of integrals for which Maple has a complete algorithm so we can make the stronger claim.

```
#--> #--> int(f3,x);

                                2
                       x exp(x ) ln(x + 1)
x - ln(x + 1) + --------------------------
                  2     2 2            2
                 x exp(x ) - ln(x + 1)

                                       2
   + 1/2 ln(ln(x + 1) + x exp(x ))

                                       2
   - 1/2 ln(ln(x + 1) - x exp(x ))
```

For definite integration problems, we can automatically invoke numerical techniques in situations that warrant it.

We first represent the problem without even attempting to evaluate it.

```
#--> r1 := Int( exp(-t) / sqrt(1-t^ 2),
#--> t = -1..1 );

                          1
                          /
                         |     exp(- t)
                 r1 :=   |  ------------- dt
                         |          2 1/2
                        /     (1 - t )
                       -1
```

We next force maple to attempt the evaluation (ie. to use `int()`).

```
#--> Eval(");

                       1
                       /
                      |   exp(- t)
                      | ------------- dt
                      |       2 1/2
                     /   (1 - t )
                    -1
```

Finally, as this did not evaluate we apply numerical techniques directly to the unevaluated result. The algebraic representation of the integrand allows for a sophisticated analysis of the numerical problem and the use of various tranformations to complete the computation.

```
#--> evalf(");

                       3.977463261
```

A platform for stepwise refinement

Computer algebra systems have a tremendous potential to support stepwise refinement. For example in Maple we can choose to examine the problem at various levels of detail.

Consider, for instance, the derivation of Simpson's rule for numerical integration (see Figure 1). Ultimately we end up with formulas like

```
#--> simpson(f(x), x=0..2,2);

             1/3 f(0) + 1/3 f(2) + 4/3 f(1)

#--> simpson(f(x), x=0..2,4);
```

```
simpson := proc( F, dx ) local a,b,f,i,h,n,rg,x;

    #   dx is an equation of the form x=a..b.
    #   n is an optional 3rd parameter
    #         for number of steps.
    x := op(1,dx); rg := op(2,dx); # dx is x=a..b
    if nargs > 2 then n := args[3] else n := 4 fi;
    a := op(1,rg); b := op(2,rg); h := (b-a)/n;
    # define a function and evaluate appropriately
    f := readlib(unapply)( F , x );
    h/3* ( f(a) + f(b)
        + 4*sum( f(a + (i*2-1)*h),i=1..n/2 )
        + 2*sum( f(a + (i*2)*h),i=1..(n/2)-1 ) );
end:
```

Figure 1: Simpson's Rule

```
  1/6 f(0) + 1/6 f(2) + 2/3 f(1/2) + 2/3 f(3/2) + 1/3 f(1)
```

or more generally,

```
#--> simpson(f(x),x=0..2,n);

2/3

                        1/2 n
                        -----
                         \        2 i - 1
  (f(0) + f(2) + 4 (      )  f(2 ---------))
                         /          n
                        -----
                        i = 1

              1/2 n - 1
                -----
                 \
       + 2 (      )        f(4 i/n)))
                 /
                -----
                i = 1

     1 / n
```

Too often, our testing consists of simply asking the students to recall this formula. But clearly, machines can compute this and the formula can be looked up. Even for our poorer students, the interest must lie in grasping the underlying mathematical model.

We need little more than the notion that a polynomial of degree two can be used to represent an arbitrary function.

We begin with three points. The chosen values of x and the corresponding values for $f(x)$ are shown in the lists below.

```
#--> xvals := [0,1,2];

                     xvals := [0, 1, 2]
```

```
#--> yvals := map(f,xvals);

                 yvals := [f(0), f(1), f(2)]
```

Under the right conditions (the ones that usually hold in class) these three points uniquely define a polynomial of degree 2. The polynomial and the function must have these three points in common. The polynomial is given by:

```
#--> interp(xvals,yvals,x);

          2                        2                         2
1/2 f(2) x  - 1/2 f(2) x - f(1) x  + 2 f(1) x + 1/2 f(0) x

   - 3/2 f(0) x + f(0)
```

We complete the derivation of Simpson's rule by integrating the resulting polynomial instead of $f(x)$.

```
#--> Int(",x=0..2);

              2                        2
Int(1/2 f(2) x  - 1/2 f(2) x - f(1) x  + 2 f(1) x

               2
   + 1/2 f(0) x  - 3/2 f(0) x + f(0),

   x = 0 .. 2)

#--> expand(");

            2                     2                2
            /                     /                /
            |   2                 |                |     2
1/2 f(2)    |  x  dx - 1/2 f(2)   |  x dx - f(1)   |  x  dx
            |                     |                |
           /                     /                /
           0                     0                0

              2                    2
              /                    /
              |                    |   2
   + 2 f(1)   |  x dx + 1/2 f(0)   |  x  dx
              |                    |
             /                    /
             0                    0

                 2
                 /
                 |
   - 3/2 f(0)    |  x dx + 2 f(0)
                 |
                /
                0

#--> Eval(");

              1/3 f(0) + 1/3 f(2) + 4/3 f(1)
```

The Maple commands form a high level description of the solution.

```
xvals := [0,1,2];
yvals := map(f,xvals);
interp(xvals,yvals,x);
```

```
Int(",x=0..2);
expand(");
Eval(");
```

Stepwise refinement is implicit in the question "What really happens when we use `interp()`?". We can proceed by defining F to be an arbitrary polynomial of degree 2,

```
#--> F := <a*x^ 2 + b*x + c|x>;

                          2
                 F := <a x  + b x + c|x>
```

We obtain three equations by evaluating both f and F at the three known points and solve the set of equations for the three unkown constants. The result is a quadratic polynomial, say R.

```
#--> eq:={f(0)=F(0), f(1)=F(1), f(2)=F(2)};

   eq := {f(0) = c, f(1) = a + b + c, f(2) = 4 a + 2 b + c}

#--> solve(eq,{a,b,c});

{c = f(0), b = - 1/2 f(2) + 2 f(1) - 3/2 f(0),

   a = 1/2 f(2) - f(1) + 1/2 f(0)}

#--> R := subs(",F(x));

                                      2
R := (1/2 f(2) - f(1) + 1/2 f(0)) x

   + (- 1/2 f(2) + 2 f(1) - 3/2 f(0)) x + f(0)
```

If we are not comfortable with solving systems of linear equations there is a simple explanation. To solve the system from first principles, choose an unsolved equation, solve it, and update all the other equations with the implied substitution. Repeat this process until done. A possible first step appears below.

```
#--> solve( f(0) = F(0) , {c} );

                        {c = f(0)}

#--> eq := subs(",eq);

eq :=

    {f(0) = f(0), f(1) = a + b + f(0),

       f(2) = 4 a + 2 b + f(0)}
```

Having derived the model, we can now observe how it behaves in specific instances using just a few additional commands.

```
#--> F := makeproc( R , x );

F := proc (x) option operator; (1/2*f(2)-f(1)+1/2*f(0))*x**2
+(-1/2*f(2)+2*f(1)-3/2*f(0))*x+f(0) end

#--> f := < x^ 4 + 3 | x>;

                            4
                   f := <x  + 3|x>

#--> plot( {f(x),F(x)}, x=0..2);
```

See Figure 2.

Additional cases may also be considered.

```
#--> f := < x + 4 | x >;
#--> plot(f(x),F(x),x=0..2);
#--> f := sin;
#--> plot(f(x),F(x),x=0..2);
```

We have been able to design the model and observe its behaviour all in the same environment.

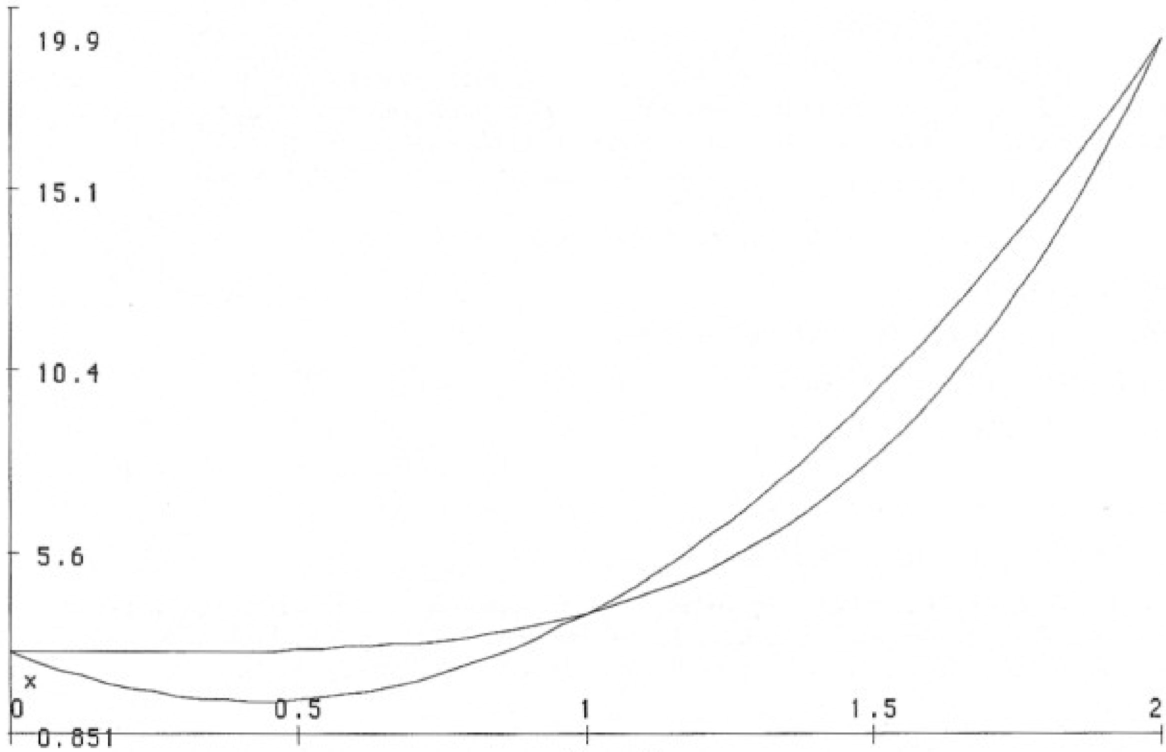

Figure 2:

Accessibility

A note is in order about user interfaces and the Macintosh release of Maple. While much of what has been discussed above is independent of the user interface, depending to a large extent on the functionality of the commands, user interfaces cannot be ignored. The easier the commands are to construct, the more likely it is that we are to try a few "computer experiments" when starting almost any new problem - the more effective our mathematical laboratory will become.

Maple: Symbolic computation for the Macintosh now runs on the Macintosh® under Finder™ (and MultiFinder™)[6] This is a complete version of Maple version 4.2 and performs well on a standard Mac Plus with only one megabyte of memory. It includes plotting and is known to run well under a variety of networking software.

The user interface of Maple on the Macintosh provides you with a *session window* (already open) to capture all the results of typical Maple commands. There are three exceptions to the use of this window for displaying results. The `help()` command displays its results in a read-only text window. Similarly the `plot()`

[6] Macintosh is a registered trademark of Apple Computer Inc. Finder and Multifinder are trademarks of Apple Computer Inc. Maple is a trademark of the University of Waterloo while Maple for the Macintosh is published by Brooks/Cole Publishing Co., Pacific Grove, Ca.

command produces its results in a special *plot window*. Finally there is a *status window* that indicates the amount of system resources you have used. See Figure 3.

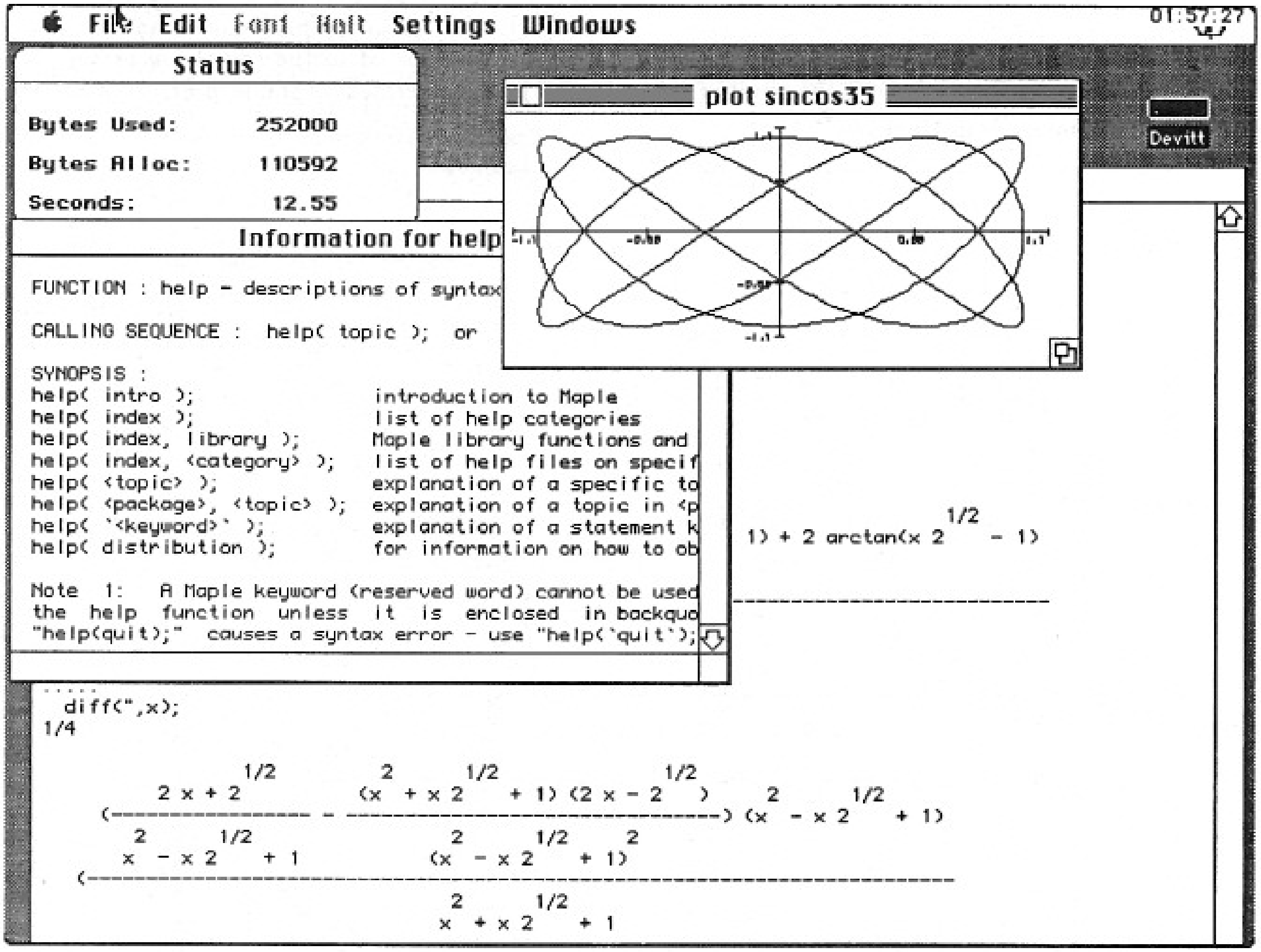

Figure 3: A Sample Screen

If you were never to venture beyond the use of this one session window you could pretend that you were using Maple on a typical large mainframe. You would simply type in your commands using the Macintosh *enter* key at the end of each line, rather than the *return* key[7]. However, a big advantage to the Mac style implementation over others is that you can edit and reuse previous commands all directly from within your Maple session.

The various windows can be thought of as multiple clipboards. Classroom examples are easily prepared and stored on a file server. After solving the problem using Maple, save the Maple instructions together with comments etc. outlining the solution process in file in the class library. By opening such files from within Maple, students have access to these documents. They may try out examples immediately by *selecting* and *entering* the associated commands. The students can easily modify the examples and save their own versions, either on diskettes or on file space provided on the server.

The standard plot command in Maple 4.2 directly supports the plotting of Maple expressions and functions, parametric plots, polar coordinates, and curves generated by lists of points. It uses adaptive techniques and is robust in that singularities, or wildly fluctuating function values are generally handled well. Several output devices are supported including regis, ln03, vt100, tektronics, postscript and ascii character plots. Several

[7] These keys have distinct functions on the Macintosh keyboard.

curves may be plotted on one graph by simply passing a set of expressions as the first argument to the plot function. All such plots may be saved as a Maple data structure, and later modified.

Maple for the Macintosh also uses a new output format for plots (`plotdevice := mac;`). By default, plots created by Maple on the Macintosh are displayed in a special *plot window*. This window can be resized, printed, or saved directly. See Figure 4. The graphs can be cut and pasted to the scrapbook or to other applications directly. Figure 5 shows the result of enhancing the graph generated by Maple in Figure 4 through a few simple commands in MacDraw®.[8]

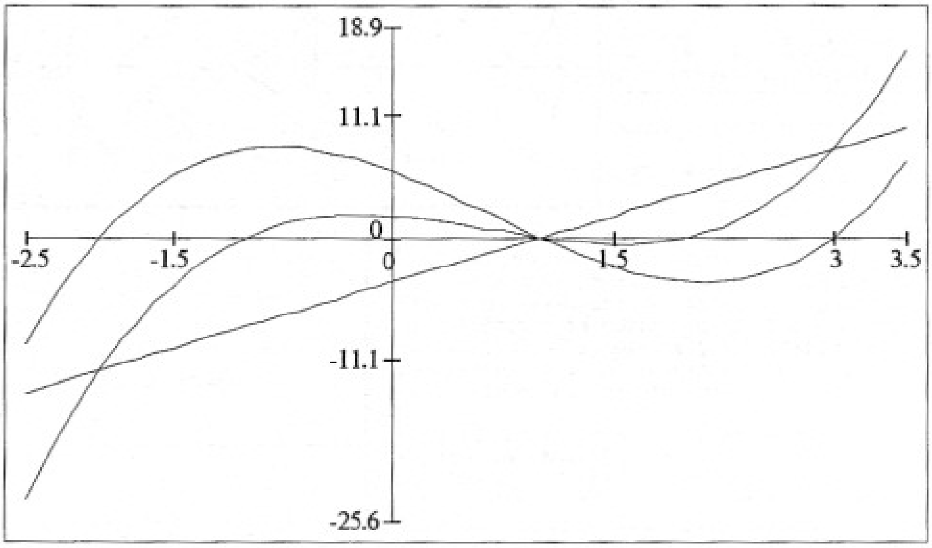

Figure 4: A Maple Plot on the Macintosh

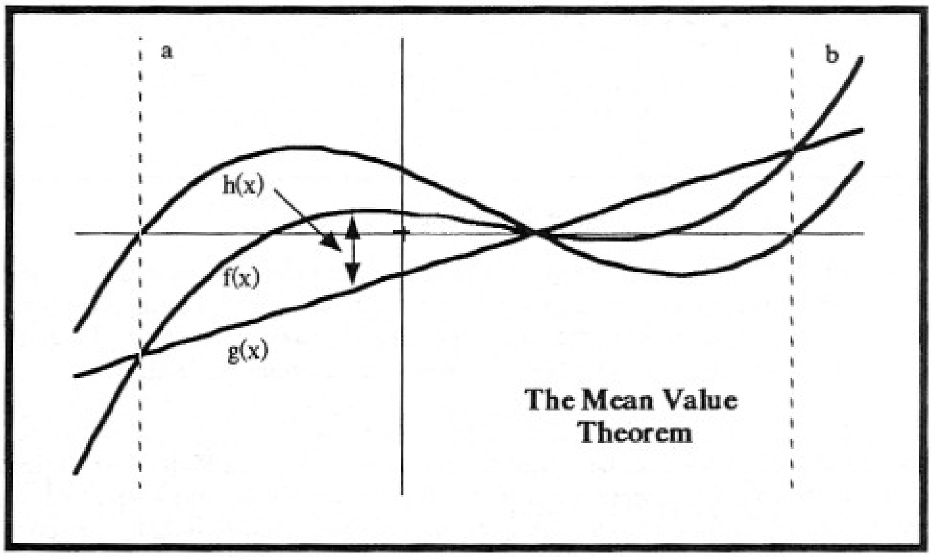

Figure 5: An Enhanced Maple Plot

Maple is small enough that it may be used effectively under MultiFinder. On a two megabyte machine, you can have Maple and at least one other application running simultaneously. Cutting and pasting between applications becomes quick and elegant.

[8] This is a registered trademark of Claris Corporation.

While much work remains to be done on the development and design of effective experimental laboratories for mathematics, all the key ingredients are in place. We are starting to address the issue of "how" a computer algebra system should be used for doing experimental mathematics. Maple's student package is a specific attempt to address this issue of style. The Maple Macintosh interface is an important first step and serves as a prototype for emerging interfaces on other work stations. The next decade promises to be an exciting time for mathematics and mathematics education.

References

[1] Char, et. al., *The Maple User's Guide* , WATCOM Publications Ltd., 415 Phillip St., Waterloo, Ontario, 1985

[2] Devitt, J. S., *Teaching First Year Calculus through the Use of Symbolic Algebra* , unpublished classroom notes, The University of Saskatchewan, October, 1987.

[3] Devitt, J. S. *The Full Power of Maple on a Mac Plus?*, the Maple Newsletter, vol 4, 1989, pp.4-9.

[4] *Toward a Lean and Lively Calculus* , MAA Notes, No. 6, edited by R. G. Douglas, The Mathematical Association of America, 1986

[5] *Calculus for a New Century: A pump not a filter*, MAA Notes, No. 8, edited by L. A. Steen, The Mathematical Association of America, 1988

[6] Hearn, Anthony C., *Reduce 2: A system and language for Algebraic Manipulation*, Proceedings of the Second Symposium on Symbolic and Algebraic manipulation, Association for Computing Machinery, 1971, pp. 128-133.

[7] Rand, R. H., *Computer Algebra in Applied Mathematics: An Introduction to MACSYMA*, Pitman Pub. Ltd. London, 1984.

[8] Ore, Oystein. *The Four Color Problem*, Academic Press, New York, 1967.

[9] Zaslove, E. *Problem E2971*, The American Mathematical Monthly, 1982.

[10] Sadowski, J. *Problem E3022*, The American Mathematical Monthly, 1983.