COMPUTER ALGEBRAIC MANIPULATION FOR THE CALCULUS OF VARIATIONS, THE MAXIMUM PRINCIPLE, AND AUTOMATIC CONTROL

BY

DAVID R. STOUTEMYER

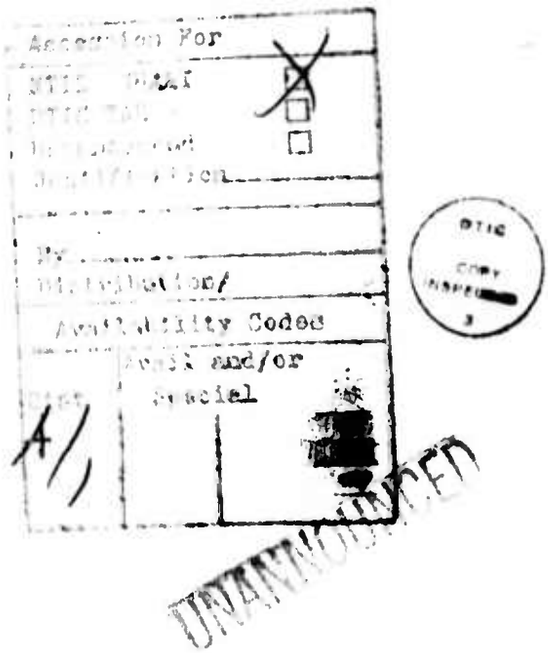
NOVEMBER 1974

TECHNICAL REPORT A74-5

COMPUTER ALGEBRAIC MANIPULATION FOR THE CALCULUS OF VARIATIONS, THE MAXIMUM PRINCIPLE, AND AUTOMATIC CONTROL

by

David R. Stoutemyer



November 1974

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 1956

*****

(-a-)

COMPUTER ALGEBRAIC MANIPULATION FOR THE CALCULUS OF VARIATIONS, THE MAXIMUM PRINCIPLE, AND AUTOMATIC CONTROL

by
David R. Stoutemyer

ABSTRACT

This report describes how to use a computer algebra system such as MACSYMA to solve variational optimization problems analytically. For a calculus of variations problem, the user merely provides expressions for the integrand of the functional and any constraints. From these expressions, a program derives the specific Euler-Lagrange equations together with first integrals when the augmented functional is free of the independent and/or any dependent variables. Other programs may then be used to construct families of analytical solutions to these differential equations and to obtain the particular solutions that satisfy given boundary conditions.

For an optimal control problem, the user provides analytical expressions for the differential constraints that govern the state variables. From these expressions, a program then determines the Hamiltonian and the differential equations that govern the auxiliary variables, together with the solutions to any trivial auxiliary equations. Other programs may then be used to search for analytical solutions to the remaining differential equations and maximize the Hamiltonian.

Examples are given illustrating how computer algebra may be used

1. to quickly and reliably derive the governing equations for large, complicated variational problems; and
2. to analytically solve the governing equations for small, simple variational problems that have closed-form analytical solutions.

Keywords and Phrases: Optimization, Calculus of variations, Maximum principle, Optimal control, Symbolic algebraic manipulation, MACSYMA

This report was supported by The ALOHA System, a research project at the University of Hawaii, which is supported by the Advanced Research Projects Agency of the Department of Defense and monitored by NASA Ames Research Center under Contract No. NAS2-8590.

1. INTRODUCTION.

A previous report by Stoutemyer [11] describes how computer algebra may be used to analytically determine the maxima, minima, and saddle points of a function of several variables -- either unconstrained or subject to equality constraints and/or inequality constraints. This report describes how computer algebra may be used to derive the differential equations that must be satisfied in order to give stationary values to a constrained or unconstrained functional of one or more independent variables and one or more dependent variables. For simple enough cases, computer algebra is also used to derive a closed-form analytical solution to these equations.

Section 2 outlines the analytical techniques that have been implemented. Section 3 is a discussion of the programming considerations for the demonstration listing in Appendix 1 and the optimization programs in Appendix 2. Section 4 summarizes some test cases. Section 5 contains conclusions and conjectures about other ways that computer algebra may be used for variational optimization. These sections may be read in any order.

2. MATHEMATICAL TECHNIQUES.

This section briefly states the mathematical principles that are implemented in the programs. There is no attempt at completeness and rigor, which can be accomplished only in a much lengthier presentation. The emphasis is on construction of candidate solutions that satisfy the necessary differential equations and boundary conditions.

Let $y = (y_1, y_2, \ldots, y_n)^T$ be a function of t, continuous with a continuous first derivative almost everywhere on $a \le t \le b$; and let y satisfy the following boundary conditions, isoperimetric constraints, and differential constraints:

$$y(a) = \tilde{y}(a), \tag{1}$$

$$y(b) = \tilde{y}(b); \tag{2}$$

$$J_j = \int_a^b f_j(t, y, y')\, dt, \qquad (j = 1, 2, \ldots, p); \tag{3}$$

$$f_j(t, y, y') = 0, \qquad (j = p+1, p+2, \ldots, q). \tag{4}$$

We seek among these admissable y, the particular ones, denoted $\tilde{y}$, that make the following functional, J_0, a local extremum:

$$J_0 = \int_a^b f_0(t, y, y')\, dt. \tag{5}$$

Here a, b, n, p, q, and J_0 through J_p are given constants.

An inequality constraint, $\phi_\mu(t, y, y') \le 0$, with $\mu > q$, may be converted to the form of (4) by adding the square of a new dependent "slack" variable, y_j, to ϕ_μ: $f_j = \phi_\mu + y_j^2 = 0$. Note also that (3) includes algebraic constraints, where y' is absent.

Denoting $(f_0, f_1, \ldots, f_q)^T$ by f and $(\lambda_0, \lambda_1, \ldots, \lambda_q)^T$ by λ, let the augmented integrand be the inner product

$$F(t, y, y', \lambda) = \lambda^T f, \tag{6}$$

where λ_0 through λ_p are unknown constants, and λ_{p+1} through λ_q are unknown functions of t. Also, let

$$F_y = \left(\frac{\partial F}{\partial y_1}, \frac{\partial F}{\partial y_2}, \ldots, \frac{\partial F}{\partial y_n}\right)^T,$$

$$F_{y'} = \left(\frac{\partial F}{\partial y_1'}, \frac{\partial F}{\partial y_2'}, \ldots, \frac{\partial F}{\partial y_n'}\right)^T,$$

where the derivatives are taken as if t, y, and y' are independent. Then $\lambda \neq 0$ and λ is continuous, except possibly where $\tilde{y}'$ is discontinuous. Also, λ and $\tilde{y}$ satisfy the DuBois-Reymond integro-differential equations:

$$F_{y'}(t, \tilde{y}, \tilde{y}', \lambda) = \int_a^t F_y(\tau, \tilde{y}, \tilde{y}', \lambda)\, d\tau + k, \tag{7}$$

where k is a vector of integration constants. Wherever $\tilde{y}'$ does not exist, it may be taken consistently as either the left or right derivative, where both are presumed to exist uniquely.

If $\tilde{y}$ and λ satisfy (7), then so do $\tilde{y}$ and any multiple of λ; so without loss of generality, we may normalize λ to 1, using any vector norm. Alternatively, when there are no differential or algebraic constraints, we may set λ_0 to any arbitrary nonzero constant, such as 1.

Wherever $\tilde{y}'$ exists, (7) may be differentiated to give the Euler-Lagrange equations:

$$\frac{d}{dt} F_{y'}(t, \tilde{y}, \tilde{y}', \lambda) = F_y(t, \tilde{y}, \tilde{y}', \lambda), \tag{8}$$

When F does not depend explicitly on t, it can be shown that

$$\tilde{y}'^{T} F_{y'}(t, \tilde{y}, \tilde{y}', \lambda) = F(t, \tilde{y}, \tilde{y}', \lambda) + k_0, \tag{9}$$

where k_0 is a constant. For many physical systems, (9) is a statement of energy conservation. Also, when F does not depend explicitly on some y_i, (7) gives

$$F_{y'_i}(t, \tilde{y}, \tilde{y}', \lambda) = k_i. \tag{10}$$

For many physical systems, (10) is a statement of the conservation of momentum or of some analagous quantity.

Another special case that occurs frequently in practice, is when F is linear in some y'_i, making the corresponding differential equation in (8) degenerate to an algebraic equation (more precisely, to an equation in finite terms).

Not treated here are well-known generalizations to more complicated boundary conditions, direct inclusion of a term $g_0[a, y(a), b, y(b)]$ outside the integral in (5), or solutions with corners. However, the program does directly treat problems with derivatives up to arbitrary order m in F, in which case (8) generalizes to

$$\sum_{j=1}^{m} (-1)^j \frac{d^j}{dt^j} F_{y^{(j)}} = F_y , \tag{11}$$

where

$$F_{y^{(j)}} = \left[\frac{\partial F}{\partial \left(\frac{d^j y_1}{dt^j}\right)}, \frac{\partial F}{\partial \left(\frac{d^j y_2}{dt^j}\right)}, \ldots, \frac{\partial F}{\partial \left(\frac{d^j y_n}{dt^j}\right)} \right]^T .$$

When J_0 is a multiple integral over an arbitrary number of independent variables, $t_1, t_2, \ldots, t_r$, the derivative on the left side of (11) generalizes to a sum over all partial derivatives of total order j. For simplicity, the program in Appendix 2 treats only the case with no mixed partial

derivatives, for which (11) generalizes to:

$$\sum_{j=1}^{m} (-1)^j \sum_{i=1}^{r} \frac{d^j}{dt^j} F_{y_{(i)}^{(j)}} = F_y , \tag{12}$$

where

$$F_y = \left[\frac{\partial F}{\partial \left(\frac{d^j y_1}{dt^j} \right)}, \frac{\partial F}{\partial \left(\frac{d^j y_2}{dt^j} \right)}, \ldots, \frac{\partial F}{\partial \left(\frac{d^j y_n}{dt^j} \right)} \right]^T .$$

The maximum principle is an approximately equivalent alternative to the calculus of variations: To convert a problem similar to (1) through (5) to a form suitable for the maximum principle:

1. Substitute an additional dependent variable, y_α, for every y_i' that occurs in f_0 in (5), and include corresponding constraints, $y_i' = y_\alpha$.

2. Wherever t occurs explicitly in (3), (4), or (5), substitute another dependent variable, y_β, and include the additional constraint $y_\beta' = 1$, together with the boundary condition $y_\beta(a) = a$.

3. For each instance of (3), introduce a unique dependent variable y_γ, and include the constraint $y_\gamma' = f_j$, together with the boundary conditions $y_\gamma(a) = 0$, $y_\gamma(b) = J_j$.

4. If a and b are considered fixed, introduce an additional dependent variable, y_ξ, together with the constraint $y_\xi' = 1$, and the boundary conditions $y_\xi(a) = a$, $y_\xi(b) = b$.

5. If J_0 includes a term $g_0[a, y(a), b, y(b)]$ outside the integral, introduce another unique dependent variable, y_η, replace f_0 with $f_0 + y_\eta$, and include the constraint $y'_\eta = 0$ together with the boundary condition $y_\eta(a) = g_0/(b-a)$.

6. Except for time-optimal problems, introduce a new variable y_0 together with the constraint $y'_0 = f_0$.

7. Differentiate any of (4) that are algebraic rather than differential constraints, and include the algebraic form evaluated at a or at b as a boundary condition.

8. Solve equations (4) simultaneously, and combine with any new constraints introduced by the above substitutions, to get all of the differential constraints in the form $y' = \tilde{f}(y)$.

The dependent variables that appear in the derivatives are called state variables, and the other dependent variables are called control or decision variables. Without loss of generality, let u denote the vector of control variables and let x denote the vector of state variables. We may then write the differential constraints or state equations as

$$x' = \tilde{f}(x, u). \tag{13}$$

Letting Ψ denote a vector of unknown time-dependent auxiliary variables with the same number of components as x, the

Hamiltonian is defined by

$$H[\psi(t), x(t), u(t)] = \psi^T \tilde{f}(x, u). \tag{14}$$

The auxiliary equations are defined by

$$\psi' = H_x, \tag{15}$$

where $H_x = (\partial H/\partial x_1, \partial H/\partial x_2, \ldots)^T$. Then the optimal control maximizes H with respect to u, while satisfying the state and auxiliary differential equations together with the boundary conditions. Moreover, the maximum value of H is nonnegative.

At maxima where $(\partial H/\partial u_1, \partial H/\partial u_2, \ldots)$ exists uniquely, it will be zero. This necessary condition may be used to help determine an analytical solution. Using Lagrange multipliers, elimination, changes of variable, and/or a combinatorial technique, this approach may be extended to the situation where u is subject to equality and/or inequality constraints, as described by Stoutemyer [11].

With the calculus of variations, an analytic solution requires

1. symbolic differentiation to derive the specific governing differential equations.

2. symbolic determination of the general closed-form solution to these differential equations.

3. symbolic integration to evaluate integrals during solution of the differential equations.

4. symbolic solution of simultaneous algebraic equations to impose the boundary conditions on the general solution.

With the maximum principle, capabilities 1 and 4 are also generally necessary to determine the maximum of the Hamiltonian.

Virtually all current algebra systems have built-in facilities for capability 1, and also for 4 when the unknowns enter linearly. Besides these, MACSYMA has built-in facilities for capability 3 and for the nonlinear case of 4, as described by Moses [7] and Yun [15] respectively. There are also two MACSYMA programs available for capability 2. One of these, written by Bogen [2], uses Laplace transforms to solve systems of arbitrary-order constant-coefficient linear ordinary differential equations. Any inhomogeneous terms are restricted to polynomial, exponential, hyperbolic, or trigonometric functions. The other program, written by Kuiper [5], uses a variety of techniques to solve linear or nonlinear first or second-order ordinary differential equations. For a brief description of a precursor to this program, see Moses [7].

3. THE PROGRAM AND EXAMPLES OF ITS USE.

MACSYMA is an interactive algebraic language with built-in capabilities for a variety of symbolic mathematical operations, including integration, differentiation, series expansion, matrix algebra, and the solution of simultaneous nonlinear equations. This language is currently available at some Honeywell 6180 MULTICS installations and on the M.I.T. MATHLAB POP-10 system. The latter system is available on the ARPANET computer network, which is described by Roberts and Wessler [8].

MACSYMA has function-definition and file-storage facilities, so that a user may write and save programs that extend the built-in capabilities. If of general interest, such user-written functions and their documentation may be placed in a public library disk file, where they are easy for others to find and use. The variational optimization functions and demonstration described here reside in such files.

MACSYMA automatically prompts the user with increasing numbered labels C1, C2, ...; and it automatically labels the corresponding output expressions D1, D2, ... When there is more than one output expression, the D-expression refers to a list of automatically generated labels beginning with the letter E.

The variational optimization file contains three functions: The function EL generates and displays one or more labeled equations, then returns a list of the labels. The equations are the Euler-Lagrange equations, perhaps together with first integrals

corresponding to conservation of energy and/or conservation of momentum. The former will contain a constant of integration K[0], whereas the latter will contain constants of integration K[i], with positive i, and will immediately follow the corresponding Euler-Lagrange equation. The function is used in the form

$$\text{EL}\ (F,\ [y_1, y_2, \ldots, y_n],\ [t_1, t_2, \ldots, t_r]),$$

where the arguments are as defined in the previous section.

The function HAM displays two or more labeled expressions, then returns a list of the labels. The first expression is the Hamiltonian, and the other expressions are the auxiliary differential equations, together with their general solutions, AUX[i] = C[i], whenever the ith differential equation is of the trivial form 'D(AUX[i],t) = 0, as is often the case. Here t is the independent variable and C[i] is the undetermined integration constant for the ith differential equation. HAM is used in the form

$$\text{HAM}\ ([\text{eqn}_1, \text{eqn}_2, \ldots]),$$

with each eqn_i of the form 'D(s,t) = expression, where s is one of the state variables and "expression" depends upon the state and control variables.

The two differential equation solvers require their differential equations in different formats. Consequently, the output of either EL or HAM is in the style required by the nonlinear equation solver ODE2, and a function CONVERT is provided to change the output to the form required by the

constant-coefficient linear differential equation solver DESOLVE. CONVERT is used in the form

CONVERT ([eqn , eqn , ...], [y , y , ...], t),
          1   2   1   2

where each eqn is an equation, each y is a dependent
             i

variable, and t is the independent variable. The output is the input equation or list of equations, with the dependencies explicitly indicated -- for example, 'D(S,T) would be replaced with 'D(S(T),T). For convenience, square brackets may be omitted from one-element list-arguments to EL, HAM, or CONVERT.

The demonstration in Appendix 1 illustrates various ways that these functions may be used. Discussion of the optimization aspects of the examples is included in the imbedded comments to make the demonstration self-contained; so that discussion is not repeated or paraphrased here. However, the comments presume an elementary knowledge of variational terminology, such as "functional", "Euler-Lagrange equation", and "Hamiltonian". The demonstration also presumes an elementary knowledge of MACSYMA or easy access to a manual. The latter assumption, while probably true of anyone who succeeds in using the demonstration, is probably not true of the majority of those who read this report. Consequently, this section contains supplementary remarks about the programming aspects of the demonstration and program. Although no attempt is made to describe any MACSYMA feature in full generality, these remarks should suffice for this

presentation. A complete description of MACSYMA is given by Bogen et. al. [3].

1. Comments, which may contain any characters except the pair */, and may appear anywhere, begin with the pair of characters /* and end with the pair of characters */.

2. Each expression typed by the user is terminated by a semicolon or dollar sign that is not within a comment. The dollar sign suppresses the printing of the D-expression, which the user may be uninterested in seeing. The user may also introduce label names of his own choosing by inserting the name followed by a colon before the rest of his input expression. Any labels may be used in subsequent expressions, where they stand for the expressions that correspond to the labels. Thus, labels fill the assignment role. For convenience, the percent symbol may be used to stand for the immediately preceding D-expression, even if printing was suppressed by use of the dollar-sign terminator for the the immediately preceding C-expression. Similarly, %TH(i) may be used for the ith preceding D-expression, with %TH(1) equivalent to %.

3. The listed computing times are in milliseconds.

4. Subscripts are enclosed in square brackets, whereas function arguments are enclosed in parentheses.

5. Not shown in Appendix 1, expressions labeled C1 through C3 were LOADFILE(OPTVAR,LISP,DSK,SHARE), LOADFILE(ODER,LISP,DSK, SHARE, and LOADFILE(DESOLN,LISP,DSK,SHARE), used to load programs needed by the demonstration. Also not shown, expression C4,

WRITEFILE(DSK,SHARE), was used to write all of the subsequent output onto a disk file so that after inserting form-feed symbols, it could be printed later on a Xerox printer, improving the appearance of the reproduction.

6. BATCH in C5 is used to read and execute the demonstration file. In practice, a user would probably use the optimization functions interactively instead, but this batch file allows the unaided user to witness a flawless demonstration, free of the inevitable typographical errors that mar an interactive listing.

7. DEPENDENCIES(Y(X)), in C8, establishes that Y depends upon X so that D(Y,X) will not evaluate to zero in the subsequent statement. The single-quote before D in C6 prevented the evaluation of the derivative at that point.

8. %TH(2)[2] in C9 refers to the second element in the second preceeding D-expression. In an interactive situation, it would be easier to use E7, but the specific label numbers are impractical to predict for a batch file, which could be loaded beginning at a point other than C5.

9. EXPAND in C10 causes the square roots to be combined.

10. SOLVE, in C11, solves the equation which is the first argument for the subexpression which is the second argument.

11. ODE2 in C13, from the file ODER LISP, solves the first or second order ordinary differential equation which is the first argument, where the second argument is the dependent variable and the third argument is the independent variable.

12. SUBST, in C14, substitutes the left side of its first argument for the right side of its first argument, in the second argument.

13. In C19, EXP denotes the exponential function, LHS denotes the left-hand side of an equation, and RHS denotes the right-hand side of an equation.

14. IC in C32, from the file ODER LISP, imposes the initial conditions indicated by its last three arguments on the general solution given by its first argument. The constants of integration in the general solution must be named K1 and K2. As seen in C34, the "initial" conditions may be imposed at either end of the interval. Not demonstrated here, an analagous function BC imposes 2-point boundary conditions, and an analagous function INITIAL1 imposes an initial condition on the general solution to a first-order differential equation with a constant of integration C.

15. C39 illustrates how statements may be grouped in parentheses, separated by commas. ATVALUE imposes the initial condition given by its third argument on its first argument, at the value of the independent variable indicated by its second argument.

16. DESOLVE in C40, from the file DESOLN LISP, solves the list of arbitrary-order simultaneous constant-coefficient linear ordinary differential equations given by its first argument, for the list of dependent variables given by its second argument. When there is only one differential equation, square brackets may be omitted from both list arguments.

17. In C67, SOLVE is used to solve the simultaneous equations given by the list which is its first argument. When there are more variables than equations, a list of the unknowns must be included as a second argument.

The remainder of this section is concerned with the definition of the variational optimization functions in Appendix 2, which may be skipped by the reader interested only in using the functions or only in learning what they can accomplish. Appendix 2 and the following remarks are included for programming enthusiasts and for reference in case anyone wishes to modify the definitions or translate them into another algebraic manipulation language.

1. In C5, the operator := defines the function HAM to be the value of the block that follows it. In the absence of a RETURN statement, the value of a block is the value of the last expression enclosed in parentheses following the word BLOCK. Expressions within a block are separated by commas, and they do not automatically cause output.

2. Enclosing variables in a list at the beginning of a block declares them local, so that they are distinct from any variables with the same names that happen to exist outside the block.

3. The LISTP function is TRUE if its argument is a list, and FALSE otherwise. Its use here permits the convenience of omitting brackets from a one-element list argument.

4. The PART function permits isolation of any term, factor, operator, etc. in a function. First it is used to isolate the second argument of the DIFF function on the left side of the first differential equation. Later it is used to isolate the first arguments of the DIFF function.

5. As used here, the LENGTH function returns the number of expressions in its list argument.

6. Similar to ALGOL, the FOR prefix iteratively executes the expression following the word DO.

7. The ENDCONS function appends its first argument to the end of the list which is its second argument.

8. In C6, PRED is used as an argument of the EV function to force the first argument to evaluate to TRUE or FALSE. Otherwise, "LT = 1" would be interpreted as an equation.

9. DERIVDEGREE returns the degree of the derivative of the second argument with respect to the third argument in the expression which is the first argument.

10. Substituting elements of the array DD for the derivatives prevents the appearance of the dependent variables in these derivatives from causing their participation in the partial derivatives with respect to the dependent variables. This technique also permits a direct test of explicit dependence upon the independent and dependent variables, so that first integrals may be constructed when appropriate.

4. TEST RESULTS.

Regarding computer analytic variational optimization, the questions of interest are: Within the available memory space and a reasonable amount of computing time,

1. what are the most complicated problems for which the programs EL or HAM can derive the Euler-Lagrange or auxiliary differential equations; and

2. of these, what are the most complicated problems for which ODE2 or DESOLVE can solve these equations?

Standard well-known computer programs and test cases are widely available for the optimization of a multivariate function, but this situation is not yet true for the optimization of functionals. Consequently, a modest literature search was conducted to find suitable test cases. The following three cases were the most complicated found:

The first case, by Stuiver [12] is concerned with transient one-dimensional compressible gas flow. A, B, and C are known constants; Y is the dependent variable; T is the time; and X is the position along the flow axis. Here is an excerpt from the program listing:

```
EL((A-'D(Y,T)-'D(Y,X)**2/2)**B*(X+T+C)**2, Y, [T,X]);
```

$$(E13)\quad \frac{D}{DX}\left(-B\,(X+T+C)^{2}\,\frac{DY}{DX}\left(-\frac{\left(\frac{DY}{DX}\right)^{2}}{2}-\frac{DY}{DT}+A\right)^{B-1}\right)$$

$$+\frac{D}{DT}\left(-B\,(X+T+C)^{2}\left(-\frac{\left(\frac{DY}{DX}\right)^{2}}{2}-\frac{DY}{DT}+A\right)^{B-1}\right)=0$$

TIME= 1697 MSEC.
(D13) [E13]

Expansion of the derivatives required another 914 milliseconds. Professor Stuiver reports that it required considerably longer than (1697 + 914) milliseconds to derive and check this result by hand. No attempt was made to derive an analytic solution to this partial differential equation.

The second case, by Payne [10] is an optimal spacecraft reentry problem. J is the functional to be minimized, X[1] through X[5] are state variables, U is the control variable, and the Ks are known constants. The list of state equations is:

$$D(J, T) = X_5\,K7 + X_4,$$

$$D(X_1, T) = -X_2\,\mathrm{SIN}(X_3),$$

$$D(X_2, T) = -\mathrm{SIN}(X_3)\,K2 - \frac{X_2^{2}\,K10\,K5\,\%E^{X_1\,K6}\,(K13\,\mathrm{SIN}(J)^{2}+K12)}{K3},$$

$$D(X_3, T) = -\frac{X_2\, K10\, K11\, K5\, \%E^{X_4 K6}\, COS(U)\, SIN(U)}{K3} + \frac{COS(X_3)\, K2}{X_2}$$

$$- \frac{X_2\, COS(X_3)}{K1 + X_1},$$

$$D(X_4, T) = X_2^3\, K4\, K5\, \%E^{\frac{X_1 K6}{2}},$$

$$D(X_5, T) = X_2^4\, K10^2\, K5^2\, \%E^{2 X_1 K6}\, K7\, (K13^2\, SIN^2(U) + K11^2\, COS^2(U)$$

$$SIN^2(U) + 2\, K12\, K13\, SIN^2(U) + K12^2)/K3 .$$

The output of HAM occupied about three pages, computed in about 6 seconds. A complete closed-form solution is hopeless; so none was attempted.

The third case, by Miele [6], is concerned with optimal nonsteady flight over a spherical earth in a great-circle plane. The physical significance of the numerous variables is unimportant here, where the purpose is merely to generate an impressively messy

set of formulas that no one but a masochist could prefer to derive by hand. The following nine expressions are constrained to equal zero:

$$D(X, T) - \frac{R\ V\ \mathrm{COS}(\mathrm{GAMMA})}{R + H},$$

$$D(H, T) - V\ \mathrm{SIN}(\mathrm{GAMMA}),$$

$$\frac{G\ R^2\ \mathrm{SIN}(\mathrm{GAMMA})}{(R + H)^2} + D(V, T) + \frac{\mathrm{DRAG}(H, V, L) - \mathrm{COS}(\mathrm{EPS})\ \mathrm{THRUST}(H, V, \mathrm{ALPHA})}{M},$$

$$D(\mathrm{GAMMA}, T) - \frac{V\ \mathrm{COS}(\mathrm{GAMMA})}{R + H} + \frac{G\ R^2\ \mathrm{COS}(\mathrm{GAMMA})}{(R + H)^2\ V} - \frac{L + \mathrm{SIN}(\mathrm{EPS})\ \mathrm{THRUST}(H, V, \mathrm{ALPHA})}{M\ V} + 2\ \mathrm{OMEGA}\ \mathrm{COS}(\mathrm{PHI}),$$

$$D(M, T) + \mathrm{BET}(H, V, \mathrm{ALPHA}),$$

$$\mathrm{THRUST}(H, V, \mathrm{ALPHA}) - \mathrm{TSI}^2,$$

$$-\mathrm{THRUST}(H, V, \mathrm{ALPHA}) + \mathrm{THRUSTMAX}(H, V) - \mathrm{ETA}^2,$$

$$A(X, H, V, \mathrm{GAMMA}, M, L, \mathrm{ALPHA}, \mathrm{EPS}),$$

$$B(X, H, V, \mathrm{GAMMA}, M, L, \mathrm{ALPHA}, \mathrm{EPS}).$$

The problem is to minimize an arbitrary function of the two end states, subject to these constraints. Using Lagrange multipliers, the output of EL occupied about five pages, computed in about 28 seconds. Here too, a complete closed-form solution is hopeless, so none was attempted. However, the computer did reveal two mistakes in the published results: the dot time-derivative operator was missing on the left side of two equations. Although these ommissions were undoubtedly typesetting rather than derivational errors, they do illustrate an advantage of direct photo-reproduction of computer-generated analytical formulas.

None of these three cases taxed the memory capacity or required an undue amount of computation time; so it would be desirable to have more complicated examples. Artificial examples could be constructed, and they have the advantage of permitting a more systematic exploration of the range of feasible problems. However, unless carefully constructed, they have the disadvantage of being non-representative of real applications. Suggestions for large real or artificial examples are welcome.

Regarding question 2 at the beginning of this section, the programs can successfully solve differential equations at least as complicated as those in Appendix 1. To develop the examples there, it required some experimentation to find simple enough coefficients and boundary conditions to make DESOLVE work. The denominator of the Laplace transform must factor into linear and quadratic factors with integer coefficients, which is unlikely when the order or number of equations exceeds 2. However, the cubic and quartic formulas are built-into MACSYMA; so the domain of the inverse Laplace transform step could easily be extended. MACSYMA also has iterative numerical routines for for finding the zeros of a polynomial to arbitrary accuracy, so at the expense of slightly inexact numerical coefficients in an analytical formula, the domain could be extended even further.

In contrast, an ODE2 demonstration file suggests that this function is capable of solving somewhat more complicated problems than those in Appendix 1. However, closed-form analytic solutions rarely exist for interesting variational problems more complicated than those in Appendix 1. Even a perfect ordinary differential equation solver, capable of finding all existing closed-form solutions, would be able to solve only a small percentage of the interesting variational problems. Consequently, no further tests were made for question 2. However, here too, suggestions for suitable examples are welcome.

5. CONCLUSIONS AND CONJECTURES.

Computer algebra is clearly helpful for avoiding the tedium and blunders associated with hand derivation of the necessary differential equations for large variational problems. It is also somewhat helpful for deriving closed-form analytical solutions when they exist.

Stoutemyer [11] describes how a similar program for optimization of a multivariate function proved to be useful as an instructional tool for an optimization course. Analytical treatment of nontrivia problems increased the students' understanding of the theoretical foundations, while nicely complementing numerical experiments. Perhaps the same will prove true for the variational optimization programs.

The examples in Appendix 1 suggest other ways that computer algebra may be used for variational optimization. For example, other necessary and sufficient conditions such as the Legendre-Clebsch, the Jacobi, the Weierstrass, and the Weierstrass-Erdmann conditions could be tested automatically. Computer algebra could also be used for a direct verification of an optimum, using techniques illustrated by Young [13,14]. Perhaps computer algebra together with artificial-intelligence theorem-proving techniques could eventually be used to automate existence and uniqueness proofs. Until then, these important questions must be answered by a separate non-computer proof or by an appeal to physical considerations.

For the calculus of variations, equation (7) has the advantage of admitting a larger class of solutions than (8), and a general-purpose program for analytically solving integro-differential equations may succeed in instances where the existing ordinary differential equation solvers fail for (7). Work has begun on such an integral equation solver, and hopefully a subsequent report will describe its successful completion.

There are also a number of ways that computer algebra could be combined with numerical methods for variational optimization:

1. The output of the functions which analytically generate the differential equations could be used as the input expressions for a standard numerical ordinary differential equation solver. The numerical routine could be written in the algebraic language, or it could be written in a language such as FORTRAN. There are facilities for generating MACSYMA output in a form suitable for direct inclusion in a FORTRAN subprogram, with common subexpressions computed separately to avoid redundant calculations.

Not demonstrated in Appendix 1, the ODER files contains routines for generating analytical series solutions by Picard iteration or Taylor series. Combined with a stepwise solution, as described by Barton and others [1] for a different computer algebra language, this comprises one of the most efficient methods for solving initial-value problems. Moreover, combined with a numerical routine for finding the zeros of a function, any

numerical initial-value routine may be used for 2-point boundary value problems, as described by Roberts and Shipman [9].

MACSYMA also has built-in facilities for Poisson series manipulation, which may be used to construct analytical perturbation-series approximations. This technique is widely used in celestial mechanics, as described by Jefferys [4].

2. The domain of the differential equation solver DESOLVE could be greatly increased by changes described in the previous section or by using various approximations of either the Laplace transform or the inverse transform.

3. Even when all of the governing differential equations cannot be completely solved analytically, computer algebra may be of use in reducirg part of the solution to quadratures and/or solving some of the equations and eliminating the corresponding variables from the remaining equations.

6. ACKNOWLEDGEMENTS.

For their help and encouragement, I thank Richard Bogen, Richard Fateman, Jeffrey Golden, Martin Griss, Ken Harrenstein, Frank Kuo, Joel Moses, Willem Stuiver, and Paul Wang. This work was made possible by grants to the MIT MATHLAB Project from the Advanced Research Projects Agency (ARPA), Department of Defense, under Office of Naval Research Contract N00014-70-A-0362-001, and to The ALOHA System, a research project at the University of Hawaii, supported by ARPA and monitored by NASA Ames Research Center under Contract No. NAS2-6700.

7. REFERENCES.

1. Barton, O., Lillers, I.M., and Zahar, R.V.M., "Taylor series methods for ordinary differential equations -- an evaluation", Rice, J.R., (editor), Mathematical Software, Academic Press, N.Y. 1971, 369-390.

2. Bogen, R.A., files DESOLN USAGE, DESOLN >, and DESOLN LISP, SHARE directory, Project MAC POP-10, MIT, Cambridge, Massachusetts, ARPANET site 198.

3. Bogen, R.A., et. al., "MACSYMA Reference Manual", The MATHLAB group, Project MAC, MIT, Cambridge, Massachusetts, 1974.

4. Jefferys, W.H., "Automated algebraic manipulation in celestial mechanics", Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, ACM, 1971, 328-331.

5. Kuiper, Ben, files OOER USAGE, OOER >, OOER LISP, OOER OEMO, and OOER OUTPUT, SHARE directory, Project MAC PDP-10, MIT, Cambridge, Massachusetts, ARPANET site 198.

6. Miele, A., "A survey of the problem of optimizing flight paths of aircraft and missles", Bellman R.E. (editor), Mathematical Optimization Techniques, Universy of California Press, Berkeley 1963, Chapter 1.

7. Moses, J., "Symbolic integration: the stormy decade", CACM 14, August 1971, 548-560.

8. Roberts, L.G., and Wessler, B., "The ARPA network", Abramson, N. and Kuo, F.F. (editors), Computer Communication Networks, Prentice-Hall Inc., Englewood Cliffs, N.J., 1973, 485-500.

9. Roberts, S.M., and Shipman, J.S., Two-Point boundary-value Problems: Shooting methods, American Elsevier Pub. Co., NY, 1972.

10. Payne, J.A., "Computational methods in optimal control problems", in Advances in Control Systems, Vol 7, Academic Press, N.Y., 1969, 73-164.

11. Stoutemyer, D.R., "Analytical optimization using computer algebraic manipulation", ALOHA technical report A74-3, University of Hawaii, 1974.

12. Stuiver, W., "On a Variational Approach to the Problems of Propagation of Plane and Spherical Waves", Ph.D. Thesis, Division of Engineering Mechanics, Stanford University, December 1959.

13. Young, L.C., Lectures on the Calculus of Variations and Optimal Control Theory, Saunders Pub. Co., Philadelphia, 1969, preamble.

14. Young, L.C., "Strengthening Caratheodory's method to apply in control problems", Balakrishnan, A.U., (editor), Control Theory and the Calculus of Variations, Academic Press, N.Y., 1969, 133-152.

15. Yun, David Y.Y., "On algorithms for solving systems of polynomial equations", SIGSAM Bulletin 27, September 1973, 19-25.

APPENDIX 1.

```
(C5) BATCH(OPTVAR, DEMO, DSK,STOUTE);
```

(C6) /* This macsyma batch file illustrates how to use the functions in OPTVAR > or OPTVAR LISP to help solve classical variational or optimal control problems. These functions use the calculus of variations and Pontryagin's Maximum-Principle to derive the governing differential and algebraic equations; and this demonstration also shows how the governing equations may be solved using the built-in SOLVE function and/or the functions in the SHARE files ODER LISP, and DESOLN LISP. For more details, see the corresponding SHARE text files OPTVAR USAGE, ODER USAGE, or DESOLN USAGE.

The illustrative examples here are intentionally elementary. For a more thorough discussion of the mathematical principles demonstrated here, and for results with more difficult examples, see the the ALOHA project technical report by David Stoutemyer, "Computer algebraic manipulation for the calculus of variations, the maximum principle, and automatic control", University of Hawaii, 1974.

Many classical variational problems are analagous to special cases of choosing a path which minimizes the transit time through a region for which the speed is a function of position. There are applications in, optics, acoustics, hydrodynamics, and routing of aircraft or ships. In the two-dimensional case, assuming the path may be represented by a single-valued function, Y(X), the transit time between Y(A) and Y(B) is given by the integral of Q(X,Y)*SQRT(1+'D(Y,X)**2), from X=A to X=B, where D is an alias for DIFF and Q(X,Y) is the reciprocal of the speed, and where we have used the fact that 'D(arclength,X) = SQRT(1+'D(Y,X)**2). For simplicity, assume Q independent of X. We may use the function EL, previously loaded by LOADFILE(OPTVAR,LISP,DSK, SHARE), to derive the associated Euler-Lagrange equation together with an associated energy and/or momentum integral if they exist: */

```
EL(Q(Y)*SQRT(1+'D(Y,X)**2), Y, X);

                                                 DY 2
                                          Q(Y)  (--)
                         DY 2                    DX
(E6)          Q(Y) SQRT((--)  + 1) - --------------- = K
                         DX                DY 2         0
                                      SQRT((--)  + 1)
                                           DX

                         DY
                 Q(Y)  --
         D               DX           DY 2       D
(E7)    -- ---------------- = SQRT((--)  + 1) (-- Q(Y))
        DX          DY 2              DX       DY
            SQRT((--)  + 1)
                    DX
TIME= 801 MSEC.
(D7)                              [E6, E7]
```

```
(C8) /* To expand the Euler-Lagrange equation: */

DEPENDENCIES(Y(X))$
TIME= 5 MSEC.

(C9) %TH(2)[2], EVAL, D;
TIME= 246 MSEC.
```

$$(D9)\quad \frac{Q(Y)\,\frac{d^2Y}{dX^2}}{\left(\frac{dY}{dX}\right)^2+1} - \frac{Q(Y)\left(\frac{dY}{dX}\right)^2\frac{d^2Y}{dX^2}}{\left(\left(\frac{dY}{dX}\right)^2+1\right)^{3/2}} = \sqrt{\left(\frac{dY}{dX}\right)^2+1}\;\left(\frac{d}{dY}\,Q(Y)\right)$$

```
(C10) /* The routines in the SHARE file ODE LISP, previously loaded,
may solve an expanded first or second order quasi-linear ordinary
differential equations; so the equation must be linear in its highest-
order derivative.  The Euler-Lagrange equation is always of this form,
but when given a second-order equation, the ODE solver often returns
with a first-order equation which we must quasi-linearize before
proceeding; so it is usually most efficient to take advantage of a
first integral when one exists, even though it requires a certain
amount of manipulation.  The SOLVE function is currently somewhat weak
with fractional powers; so we must massage the above energy integral
before solving for 'D(Y,X): */

%TH(3)[1] * SQRT(1+'D(Y,X)**2), EXPAND, EVAL;
TIME= 211 MSEC.
```

$$(D10)\qquad Q(Y) = K_0\left(\frac{dY}{dX}\right)^2+1$$

```
(C11) SOLVE(%/K[0], 'D(Y,X));
SOLUTION
```

$$(E11)\qquad \frac{dY}{dX} = -\frac{\sqrt{Q(Y)^2 - K_0^2}}{K_0}$$

$$(E12)\qquad \frac{dY}{dX} = \frac{\sqrt{Q(Y)^2 - K_0^2}}{K_0}$$

```
TIME= 1591 MSEC.
(D12)                          [E11, E12]
```

```
(C13) ODE2(EV(%[2],EVAL), Y, X);

TIME= 10722 MSEC.
```

$$(D13)\qquad X - K_0 \left[\int \frac{1}{\sqrt{Q(Y)^2 - K_0^2}}\,dY\right] = C\,K_0$$

```
(C14) /* We must specify Q(Y) to proceed further.  Q(Y)=1 is associat-
ed with the Euclidian shortest path (a straight line), Q(Y)=SQRT(Y) is
associated with the stationary Jacobian-action path of a projectile (a
parabola), Q(Y)=Y is associated with the minimum-surface
body of revolution (a hyperbolic cosine), and Q(Y)=1/SQRT(Y)
is associated with the minimum-time path of a falling body, starting
at Y=0, measured down, (a cycloid).  Of these, the cycloid presents
the most difficult integration.  In fact, I have never seen
the integration for this case performed directly except by macsyma: */

SUBST(Q(Y)=1/SQRT(Y), %) $
TIME= 124 MSEC.

(C15) %,INTEGRATE;
TIME= 4881 MSEC.
```

$$(D15)\quad -\left(X - K_0\left(\frac{\sqrt{\frac{1}{Y} + K_0^2}}{2\,K_0^4\left(\frac{1}{Y} + K_0^2\right) - K_0^4} + \frac{\log\left(\sqrt{\frac{1}{Y} + K_0^2} - K_0\right)}{2\,K_0^3}\right)\right)/K_0 - \frac{\log\left(\sqrt{\frac{1}{Y} + K_0^2} + K_0\right)}{2\,K_0^3}\Bigg)\Big)/K_0 = C$$

```
(C16) /* This equation may be simplified by combining the LOGs and
clearing some fractions, but it is transcendental in Y; so there is no
hope for a closed-form representation for Y(X).  For completeness we
should try the other alternative for 'D(Y,X), but it turns out to
lead to the same result.

Most authors solve this problem by introducing the change of variable
'D(Y,X)=TAN(T), useful also for other Q(Y), which leads to the para-
metric representation: Y=K[0]*COS(T)**2, X=C+K[0]*(T+SIN(T)*COS(T)).
Solving the former for T and substituting into the latter gives the
representation equivalent to that found directly by macsyma.
```

Some straightforward investigation reveals that C is the initial value of X, but the equation is transcendental in K[0], so K[0] cannot be determined analytically. However, it may be determined to arbitrary numerical accuracy by an iterative algorithm such as that described in the SHARE file ZEROIN USAGE.

To prove that the above solution is a strong global minimum, we must prove that such a minimum exists and that no other answer gives a smaller value to the functional. This may be done by the direct approach of Caratheodory, or by checking the classical Weierstrass, Weierstrass-Erdman, Legendre, and Jacobi necessary and sufficient conditions. Computer algebra can assist these investigations, but they are beyond the scope of this demonstration. Instead, to keep the demonstration brief, we will now consider other kinds of problems.

Stationary values of a functional may be subject to algebraic or differential constraints of the form F(X,Y(X),'D(Y,X))=0 and/or isoperimetric constraints of the form INTEGRATE(F(X,Y(X),'D(Y,X)),X,A,B) = J, where J is a given constant. Sometimes it is possible to eliminate one or more constraints, substituting them into the functional and any remaining constraints; but if not, we may use Lagrange multipliers. For example, suppose we wish to determine the curve that a string of given length assumes to minimize its potential energy. The potential energy is INTEGRATE(Y*SQRT(1+'D(Y,X)**2), X, A, B), whereas the length is INTEGRATE(SQRT(1+'D(Y,X)**2), X, A, B); so letting MULT be the Lagrange multiplier, our augmented functional is of the form that we have been studying, with Q(Y)=Y+MULT. Consequently, we may simply substitute this in the general integral that we found before: */

```
SUBST(Q(Y)=Y+MULT, %TH(3))$
TIME= 117 MSEC.

(C17) /* Type  NONZERO;  in response to the question generated by
the following statement: */

%,INTEGRATE;
IS  K    ZERO OR NONZERO?
     0

NONZERO;
TIME= 1338 MSEC.
```

$$(D17) \qquad X - K_0 \, \mathrm{LOG}\left(\mathrm{SQRT}\left(\frac{(Y + \mathrm{MULT})^2}{K_0^2} - 1\right) + \frac{Y + \mathrm{MULT}}{K_0}\right) = C$$

```
(C18) /* To obtain Y as a function of X: */

(%*K[0]+X)/K[0];
TIME= 92 MSEC.
```

$$\text{(D18)}\qquad \log\left(\sqrt{\frac{(Y+\text{MULT})^2}{K_0^2}-1}+\frac{Y+\text{MULT}}{K_0}\right)=\frac{X+K_0\,C}{K_0}$$

```
(C19) EXP(LHS(%))=EXP(RHS(%));
TIME= 28 MSEC.
```

$$\text{(D19)}\qquad \sqrt{\frac{(Y+\text{MULT})^2}{K_0^2}-1}+\frac{Y+\text{MULT}}{K_0}=\%E^{\frac{X+K_0\,C}{K_0}}$$

```
(C20) SOLVE((%*K[0]-Y-MULT)**2, Y);
SOLUTION
```

$$\text{(E20)}\qquad Y=\frac{\%E^{-\frac{X}{K_0}-C}-\left(K_0\,\%E^{\frac{2X}{K_0}+2C}-2\,\%E^{\frac{X}{K_0}+C}\,\text{MULT}+K_0\right)}{2}$$

```
TIME= 7851 MSEC.
(D20)                                [E20]

(C21) %,EVAL,EXPAND,RATPRODEXPAND:TRUE;
TIME= 17 MSEC.
```

$$\text{(D21)}\qquad Y=\frac{K_0\,\%E^{-\frac{X}{K_0}-C}}{2}+\frac{K_0\,\%E^{\frac{X}{K_0}+C}}{2}-\text{MULT}$$

```
(C22) /* We may rewrite the above expression as: */

K[0]*COSH(X/K[0]+C) - MULT $
TIME= 2457 MSEC.
```

(C23) /* We may substitute this into the integral constraint to get 1 equation relating the constant K[0] and C to the given length L: */

```
SQRT(1+D(%,X)**2);
TIME= 96 MSEC.
                                     2 X
(D23)                      SQRT(SINH (-- + C) + 1)
                                      K
                                       0
```

(C24) /* This is simply COSH(X/K[0]+C), so: */

```
L = INTEGRATE(COSH(X/K[0]+C), X, A, B);

TIME= 1787 MSEC.
                           K  C + B            K  C + A
                            0                   0
(D24)            L = K  SINH(--------) - K  SINH(--------)
                      0         K         0         K
                                 0                   0
```

(C25) /* Given A, B, Y(A), and Y(B), we may substitute into the general solution to get two more relations among K[0], C and MULT, but the three equations are transcendental; so a numerical solution is generally necessary.

For completeness, we should also investigate the case of K[0]=0, which yields the solution Y=0; but for brevity, we will omit that analysis.

The functional may include derivatives of higher than first order. For example, including the small-deflection energy of bending, shear, and an elastic foundation, the elastic energy of a nonuniform beam with loading W(X) is the integral of the following function: */

```
A(X)*'D(Y,X,2)**2 + B(X)*'D(Y,X)**2 + C(X)*Y**2 + W(X)*Y$
TIME= 97 MSEC.
```

(C26) /* To obtain the Euler-Lagrange equation: */

```
EL(%, Y, X);

                                   2           2
                 D        DY     D           D Y
(E26)            -- 2 B(X) -- - --- 2 A(X) --- = 2 C(X) Y + W(X)
                 DX       DX      2           2
                                DX          DX
TIME= 918 MSEC.
(D26)                                 [E26]
```

(C27) /* If shear and foundation energy are negligible, as is often the case, we may solve this differential equation, for specific A(X) and W(X), by two successive applications of ODE2. For example: */

```
%[1], EVAL,A(X)=X, B(X)=0, C(X)=0, W(X)=1, 'D(Y,X,2)=DY2$
TIME= 336 MSEC.

(C28) DEPENDENCIES(DY2(X))$
TIME= 4 MSEC.

(C29) EV(%TH(2),D,EVAL);
TIME= 159 MSEC.
```

$$(D29) \qquad -2\,\frac{d^2y}{dx^2}\,x - 4\,\frac{dy}{dx} = 1$$

```
(C30) ODE2(%, DY2, X);

YOU HAVE RUN OUT OF LIST SPACE.
DO YOU WANT MORE?
TYPE ALL; NONE; OK; A LEVEL-NO. OR THE NAME OF A SPACE.
1;
TIME= 3720 MSEC.
```

$$(D30) \qquad DY2 = -\frac{X}{4} + \frac{K2}{X} - \frac{K1}{2}$$

```
(C31) ODE2(SUBST([K1=K3,K2=K4,DY2='D(Y,X,2)],%), Y, X);
TIME= 1959 MSEC.
```

$$(D31) \qquad Y = -\frac{K4\,X\,(24 - 24\,LOG(X)) + X^3 + 6\,K3\,X^2}{24} + K2\,X + K1$$

```
(C32) /* Now let's impose the boundary conditions  Y='D(Y,X)=0 at X=1,
Y=0, 'D(Y,X)=1 at X=2: */

IC(%, X=1, Y=0, 'D(Y,X)=0);
TIME= 1604 MSEC.
```

$$(D33) \qquad [Y = -\frac{K4\,X\,(24 - 24\,LOG(X)) + X^3 + 6\,K3\,X^2}{24} + \frac{(4\,K3 + 1)\,X}{8} + \frac{12\,K4 - 3\,K3 - 1}{12}]$$

```
(C34) IC(SUBST([K3=K1,K4=K2],FIRST(%)), X=2, Y=0, 'D(Y,X)=1);
TIME= 3068 MSEC.
```

$$(D35)\quad [Y = -\frac{\frac{49\,X\,(24 - 24\,\log(X))}{72\,\log(2) - 48} + X^3 - \frac{6\,(110\,\log(2) - 57)\,X^2}{18\,\log(2) - 12}}{24} + \frac{(1 - \frac{4\,(110\,\log(2) - 57)}{18\,\log(2) - 12})\,X}{8} + \frac{-\frac{588}{72\,\log(2) - 48} + \frac{3\,(110\,\log(2) - 57)}{18\,\log(2) - 12} - 1}{12}]$$

(C36) /* as another specific beam example: */

```
%TH(8),EVAL, A(X)=1,B(X)=2,C(X)=1,W(X)=1$
TIME= 268 MSEC.
```

(C37) %[1],D;
TIME= 179 MSEC.

$$(D37)\quad 4\,\frac{d^2Y}{dX^2} - 2\,\frac{d^4Y}{dX^4} = 2\,Y + 1$$

(C38) /* We cannot treat this as 2 successive second-order equations, but the function DESOLVE in the SHARE file DESOLN, already loaded, is applicable to some linear arbitrary-order constant coefficient equations. First we must convert the equation so that the dependency of Y is explicitly indicated: */

```
CONVERT(%, Y, X);
TIME= 215 MSEC.
```

$$(D38)\quad 4\,\left(\frac{d^2}{dX^2}\,Y(X)\right) - 2\,\left(\frac{d^4}{dX^4}\,Y(X)\right) = 2\,Y(X) + 1$$

(C39) /* DESOLVE requires all boundary conditions to be at X=0, and it works best if they are specified before calling the function. However, we may overcome this restriction, as illustrated by the following example, where the beam has zero slope and deflection at both ends: */

```
(ATVALUE(Y(X),X=0,0), ATVALUE('D(Y(X),X),X=0,0),
      ATVALUE('D(Y(X),X,2),X=0,K1), ATVALUE('D(Y(X),X,3),X=0,K2))$
TIME= 78 MSEC.
```

```
(C40) DESOLVE(%TH(2), Y(X));

TIME= 2857 MSEC.
```

$$(D40)\quad Y(X) = \frac{(2K2 - 2K1 + 1)\, X\, \%E^{-X}}{8} + \frac{(K2 + 1)\, \%E^{-X}}{4} + \frac{(2K2 + 2K1 - 1)\, X\, \%E^{X}}{8} - \frac{(K2 - 1)\, \%E^{X}}{4} - \frac{1}{2}$$

```
(C41) IC(SUBST(Y(X)=Y,%), X=1, Y=0, 'D(Y,X)=0);
TIME= 7259 MSEC.
```

$$(D42)\quad [Y = \frac{\left(\frac{2\,(\%E^2 - 2\,\%E - 1)}{2\,\%E^2 + 4\,\%E - 2} + \frac{2\,(\%E^2 - 2\,\%E + 1)}{\%E^2 + 2\,\%E - 1} + 1\right) X\, \%E^{-X}}{8} + \frac{\left(\frac{\%E^2 - 2\,\%E + 1}{\%E^2 + 2\,\%E - 1} + 1\right) \%E^{-X}}{4} + \frac{\left(-\frac{2\,(\%E^2 - 2\,\%E - 1)}{2\,\%E^2 + 4\,\%E - 2} + \frac{2\,(\%E^2 - 2\,\%E + 1)}{\%E^2 + 2\,\%E - 1} - 1\right) X\, \%E^{X}}{8} + \frac{\left(\frac{\%E^2 - 2\,\%E + 1}{\%E^2 + 2\,\%E - 1} - 1\right) \%E^{X}}{4} - \frac{1}{2}]$$

```
(C43) /* We may also treat problems with more than one dependent
variable.  For example, the charge, Q, and displacement from
equilibrium, Y, of a simple electromagnetic loudspeaker, with M, K,
L, S, E(T), and T denoting mass, spring constant, inductance, electro-
magnetic work coefficient, voltage, and time, respectively, are given
by the stationary value of the integral of the function
in the first argument of EL below.  (ref. S.H. Crandall, D.C. Karnop,
E.F. Kurtz Jr., & D.C. Pridmore-Brown, "Dynamics of Mechanical and
Electromechanical Systems", McGraw-Hill), */
```

```
EL((M*'D(Y,T)**2 - K*Y**2 + L*'D(Q,T)**2)/2 + S*'D(Q,T)*Y + E(T)*Q,
   [Y, Q], T);
```

$$(E43)\qquad \frac{D}{DT}\left(M\,\frac{DY}{DT}\right) = \frac{DQ}{DT}\,S - K\,Y$$

$$(E44)\qquad \frac{D}{DT}\left(S\,Y + L\,\frac{DQ}{DT}\right) = E(T)$$

```
TIME= 4291 MSEC.
```

$$(D44)\qquad [E43,\ E44]$$

```
(C45) /* We may simplify these equations by replacing 'D(Q,T) with
the current, I; and we may introduce linear mechanical resistance B
and linear electrical resistance R as follows: */

%, EVAL, 'D(Q,T)=I, S*I=S*I-B*'D(Y,T), E(T)=E(T)-R*I;
TIME= 471 MSEC.
```

$$(D45)\qquad \left[\frac{D}{DT}\left(M\,\frac{DY}{DT}\right) = -B\,\frac{DY}{DT} - K\,Y + I\,S,\ \frac{D}{DT}\left(S\,Y + I\,L\right) = E(T) - I\,R\right]$$

```
(C46) /* DESOLVE  also works for some systems of arbitrary-order
constant-coefficient equations; so let's try it with the following
parameter values, excitation E(T), and initial conditions: */

SUBST([M=2,K=1,B=1,S=1,L=1,E(T)=SIN(T),R=1], %)$
TIME= 483 MSEC.

(C47) CONVERT(%, [Y,I], T)$
TIME= 461 MSEC.

(C48) (ATVALUE(Y(T),T=0,0), ATVALUE(I(T),T=0,0), ATVALUE('D(Y(T),T),T=
0,0))$
TIME= 52 MSEC.

(C49) DESOLVE(%TH(2), [Y(T),I(T)]);
TIME= 10219 MSEC.
```

$$(D50)\qquad [Y(T) = \%E^{-T/2}\left(\frac{\sin\left(\frac{\sqrt{3}\,T}{2}\right)}{\sqrt{3}} - \frac{\cos\left(\frac{\sqrt{3}\,T}{2}\right)}{3}\right) - \frac{2\sin(T)}{5} - \frac{\cos(T)}{5} + \frac{8\,\%E^{-T/2}}{15},$$

$$I(T) = \%E^{-T/2}\left(-\frac{\sin\left(\frac{\sqrt{3}\,T}{2}\right)}{\sqrt{3}} - \frac{\cos\left(\frac{\sqrt{3}\,T}{2}\right)}{3}\right) + \frac{3\sin(T)}{5} - \frac{\cos(T)}{5} + \frac{8\,\%E^{-T/2}}{15}$$

(C51) /* The functional may also have more than 1 independent variable. For example, the general 2-dimensional linear elliptic partial differential equation is equivalent to the variational problem:*/

```
EL(A*'D(U,X)**2 + B*'D(U,Y)**2 + C*U**2 + E*'D(U,X) + F*'D(U,Y) +
   G*U,  U,  [X,Y]);
```

$$(E51)\qquad \frac{D}{DY}\left(2\,B\,\frac{DU}{DY} + F\right) + \frac{D}{DX}\left(2\,A\,\frac{DU}{DX} + E\right) = 2\,C\,U + G$$

```
TIME= 1163 MSEC.
(D51)                                [E51]
```

(C52) /* Analytic solutions to even the simplest cases of this equation are rare, but computer algebraic manipulation has been used to construct series solutions to such equations.

Many variational optimization problems are easier to treat using Pontryagin's Maximum-Principle than by the calculus of variations. This is particularly true of optimal control problems.

As an elementary example, suppose that we have a unit mass at an arbitrary position X[0] with arbitrary velocity V[0] at time T=0, and we wish to vary the force F, subject to the constraint -1 <= F <= 1, such that the mass arrives at position X=0 with velocity V=0, in minimum time. The force equals the rate of change of momentum; so the motion is governed by the pair of first-order differential equations: */

```
['D(V,T)=F,
'D(X,T)=V]$
TIME= 24 MSEC.
```

(C53) /* Using this list as the argument to the function HAM results in output of the Hamiltonian followed by differential equations for the so-called Auxiliary variables, together with their solution whenever the differential equation is of the trivial form:
'D(aux[i],t) = 0, as is often the case: */

```
HAM(%);
```

(E53)
$$AUX_2\ V + AUX_1\ F$$

(E54)
$$\frac{D}{DT}\ AUX_1 = -\ AUX_2$$

(E55)
$$\frac{D}{DT}\ AUX_2 = 0$$

(E56)
$$AUX_2 = C_2$$

```
TIME= 341 MSEC.
```
(D56) [E53, E54, E55, E56]

```
(C57) /* We may substitute the given solution of the last
differential equation into the one before it, then use either DESOLVE
or ODE2: */

%[4],EVAL$
TIME= 25 MSEC.

(C58) ODE2(EV(%TH(2)[2],%,EVAL), AUX[1], T);
TIME= 464 MSEC.
```
(D58)
$$AUX_1 = C_1 - C_2\ T$$

```
(C59) /* Now we may substitute the values of the auxiliary variables
into the Hamiltonian: */

%TH(3)[1], %, EVAL$
TIME= 125 MSEC.

(C60) SUBST(%TH(3), %);
TIME= 35 MSEC.
```
(D60)
$$C_2\ V + F\ (C_1 - C_2\ T)$$

```
(C61) /* According to the maximum principle, we should vary F so as to
maximize this expression for all values of T in the time-interval
of interest.  Considering the constraints on  F, clearly F should be
SIGN(C-C[2]*T).  (We could use the function  STAP in
the SHARE file OPTMIZ >  or OPTMIZ LISP  to determine this.)

We have found that every optimal control is  F=1  and/or F=-1, with at
most one switch between them, when T=C/C[2].  Since  F  is piecewise
constant, we may combine the two first-order equations of motion and
solve them as follows: */
```

```
ODE2('D(X,T,2)=F, X, T);
TIME= 617 MSEC.
```

$$\text{(D61)} \qquad X = \frac{F\ T^2}{2} + K2\ T + K1$$

(C62) /* Now we may choose the constants of integration so as to satisfy any given boundary conditions. For example, suppose that we start with V=0 and X=1 at T=0. Assume we start with F=-1: */

```
IC(SUBST(F=-1,%), T=0, X=1, 'D(X,T)=0);
TIME= 634 MSEC.
```

$$\text{(D63)} \qquad [X = 1 - \frac{T^2}{2}]$$

(C64) /* Assuming that we terminate with F=+1: */

```
IC(SUBST(F=1,%TH(2)), T=TFINAL, X=0, 'D(X,T)=0);
TIME= 3938 MSEC.
```

$$\text{(D65)} \qquad [X = \frac{TFINAL^2}{2} - T\ TFINAL + \frac{T^2}{2}]$$

(C66) /* To determine TFINAL and the time T at which F switches sign, we may impose the condition that the two solutions must agree in position and velocity at that time: */

```
SUBST(%, FIRST(%TH(2)));
TIME= 36 MSEC.
```

$$\text{(D66)} \qquad \frac{TFINAL^2}{2} - T\ TFINAL + \frac{T^2}{2} = 1 - \frac{T^2}{2}$$

```
(C67) SOLVE([%, D(%,T)]);

TIME= 3590 MSEC.
(D67)     [[TFINAL = 2.0, T = 1.0], [TFINAL = - 2.0, T = - 1.0]]
```

(C68) /* For the first of these solutions, 0 < T < TFINAL; so the assumption that F switches from -1 to 1 is justified. F is now completely determined as a function of time, but to represent it with our expression SIGN(C-C[2]*T): */

```
SOLVE(SUBST(%[1], C-C[2]*T), C);
SOLUTION
```

$$\text{(E68)} \qquad C = C_2$$

```
TIME= 99 MSEC.
(D68)                                  [E68]
```

```
(C69) /* As is always the case, one of the integration constants for
the auxiliary equations is redundant; so we may set C[2] and C to the
same arbitrary negative constant, such as -1.

It is important to remember that so far, we have only
determined an EXTREMAL control.  To prove that it is an OPTIMAL
control, we must prove that an optimal control exists and that no
more-optimal extremal control exists.  However, such considerations
are beyond the scope of this demonstration.  */

TIME= 161874 MSEC.
(D69)                          BATCH DONE
```

APPENDIX 2.

```
(C3) /* This file contains functions and option settings for
variational optimization using the calculus of variations and
the maximum principle.  For a description of its usage see the text
file  OPTVAR USAGE. */

/* Set options to automatically print cpu time in milliseconds, force
attempted equation solution even when there are more variables than
unknowns, when an equation involves logs or exponentials, or when a
coefficient matrix is singular: */

TIME:GRINDSWITCH:SOLVERADCAN:SINGSOLVE:TRUE$
TIME= 13 MSEC.

(C4) /* establish  D  as an alias for the differentiation function: */

ALIAS(D,DIFF)$
TIME= 5 MSEC.

(C5) HAM(ODES) := BLOCK(
/* This function computes the Hamiltonian & the auxiliary equations */

   [T,NSV,STATEVARS,AUXVARS,ANSW,ELIST,AUXDE], /*declare local vars*/

   IF NOT LISTP(ODES) THEN ODES: [ODES], /* ensure list argument */
   T: PART(ODES,1,1,2),  /* get independent var from derivative */
   NSV: LENGTH(ODES),  /* determine number of state variables */
   /* Form list of state and auxiliary variables: */
   STATEVARS: AUXVARS: ELIST: [],
   FOR I THRU NSV DO (STATEVARS: ENDCONS(PART(ODES,I,1,1), STATEVARS),
      AUXVARS: ENDCONS(AUX[I], AUXVARS)),
   ANSW: [SUM(RHS(ODES[I])*AUX[I], I, 1, NSV)], /* form Hamiltonian */

   /* Form list of auxiliary equations and any trivial solutions: */
   FOR I THRU NSV DO (
      AUXDE: 'DIFF(AUX[I],T) = -DIFF(ANSW[1], STATEVARS[I]),
      ANSW: ENDCONS(AUXDE, ANSW),
      IF RHS(AUXDE)=0 THEN ANSW:ENDCONS(AUX[I]=C[I],ANSW)),

   /* Form list of E-labels while displaying computed results: */
   FOR ITEM IN ANSW DO ELIST: ENDCONS(FIRST(DISP(ITEM)), ELIST),
   ELIST) $
 TIME= 102 MSEC.
```

```
(C6) EL(F, YLIST, TLIST) := BLOCK( /* This function computes the
Euler-Lagrange equations and any trivial first integrals: */

   [LY,LT,FSUB,ENERGYCON,ANSW,ELIST],  /* declare local variables */

   IF NOT LISTP(TLIST) THEN TLIST: [TLIST],
   IF NOT LISTP(YLIST) THEN YLIST: [YLIST],
   /* compute number of independent & independent variables: */
   LY: LENGTH(YLIST), LT: LENGTH(TLIST), FSUB: F,


   /* no conservation of energy if more than 1 independent var: */
   ENERGYCON: EV(LT=1,PRED),
   FOR I THRU LY DO /* substitute for derivatives: */
      FOR J THRU LT DO (DD[I,J]: DERIVDEGREE(FSUB,YLIST[I],TLIST[J]),
         IF DD[I,J] > 1 THEN ENERGYCON: FALSE,
         FOR K THRU DD[I,J] DO
            FSUB:SUBST('DIFF(YLIST[I],TLIST[J],K)=DYDT[I,J,K], FSUB)),
   /* no conservation of energy if independent var. in integrand: */
   IF NOT FREEOF(TLIST[1],FSUB) THEN ENERGYCON: FALSE,
   ANSW: IF ENERGYCON THEN [FSUB] ELSE [], /* form list of results: */
   FOR I THRU LY DO (FY: DIFF(FSUB,YLIST[I]),
      ANSW: ENDCONS(
         SUM(SUM((-1)**(K-1)*'DIFF(DIFF(FSUB,DYDT[I,J,K]),TLIST[J],K),
         K,1,DD[I,J]), J, 1, LT) = FY, ANSW),
      IF ENERGYCON THEN ANSW[1]: ANSW[1] -
         DIFF(FSUB,DYDT[I,1,1])*'DIFF(YLIST[I],TLIST[1]),
      IF FY=0 AND LT=1 AND DD[I,1]=1 THEN /* momentum integral */
         ANSW: ENDCONS(DIFF(FSUB,DYDT[I,1,1])=K[I], ANSW)),
   IF ENERGYCON THEN ANSW[1]: ANSW[1]=K[0],

   FOR I THRU LY DO  /* resubstitute original derivatives: */
      FOR J THRU LT DO
         FOR K THRU DD[I,J] DO
            ANSW:SUBST(DYDT[I,J,K]='DIFF(YLIST[I],TLIST[J],K), ANSW),

   ELIST:[],  /* form list of E-labels while displaying results: */
   FOR EQN IN ANSW DO ELIST: ENDCONS(FIRST(DISP(EQN)), ELIST),
   ELIST) $
TIME= 194 MSEC.

(C7) CONVERT(ODES, YLIST, T) := BLOCK([ANSW],
/* This function converts output of EL or HAM to form required by
OESOLVE  from the file  DESOLN. */
   IF NOT LISTP(YLIST) THEN YLIST: [YLIST],
   ANSW: EV(ODES,EVAL),  /* if E-labels, replace with values */
   FOR YY IN YLIST DO ANSW: SUBST(YY=YY(T), ANSW),
   RETURN(ANSW)) $
TIME= 29 MSEC.
```