Ph20: One dimensional Motion with Drag

Lecture Notes Ph.20 Fox
One Dimensional Motion with Drag

I.2: One Dimensional Motion with Drag

A Physics References:

Many interesting problems Sections 5.1 to 5.5

[Bennett 76]

Rockets:

p. 186,205 [Eisberg 81A]

Viscous Fluids:

p. 150 [Eisberg 81A]

p. 400 [Roller 81]

Skydiver

p. 190, 204 [Eisberg 81A]

B Numerical Analysis References:

Interpolation:

Section 20.6 [Roller 81]

Chapter 3 [Burden 81]

Integration of Ordinary Differential Equations.

Section 5.3 [Bennett 76]

Sections 6.6, 13.5 [Roller 81]

Chapter 5 [Burden 81]

Solution of Equations in One Variable

Section 21.6 [Roller 81]

Chapter 2 [Burden 81]

C Introduction:

The program in ONEDIM1.BAS can be used to address problems in Bennett 76, page 211-213, Section 5.5

Note that we use units of metres and not feet as in the book and so the relevant parameters are:

Gravity a_o = 9.797 m sec^{-2}

and the air resistance numbers are

$$
\begin{array}{lll}
a_2(\text{fetal}) & .0017 & m^{-1} \\
a_2(\text{nosedive}) & .0025 & m^{-1} \\
a_2(\text{horizontal}) & .0034 & m^{-1}
\end{array}
\qquad (1)
$$

We solve the differential equation

$$\ddot{y} = a_o - a_2\,\dot{y}\,|\dot{y}\,| \qquad (2)$$

where y measures the vertical height. y increases as one goes downwards.

We convert this to a set of 2 first order equations for ξ_1 = y and $\xi_2 = \frac{dy}{dt}$

$$\dot{\xi}_1 = \xi_2 \qquad (3)$$

$$\dot{\xi}_2 = a_o - a_2\,\xi_2|\xi_2|$$

D: Use of Program

In reply to:

Drag coefficient a2?

enter the appropriate value 0 will give motion without drag

.0034ENTER

will give the Skydiver her maximum resistance

In reply to.

Initial T, Y, dy/dt?

enter the initial values of time, height, velocity,

t_o, y_o, v_o separated by commas, for instance

o,o,o ENTER

In reply to

T or Y range?

use either

y ENTER

or

t ENTER

If response was y, program will integrate from y_o to a final value of $y = y_f$ given by response to

Final Y value?

76.2 ENTER

is an appropriate reply for problem 5.2 of Bennett 76 (p 211)

If response was t to "T or Y range?", the program will fix not the y interval but the t interval. In this case, one integrates from t_o to t_f given by the response to

Final T value?

Now the problem is set up, the user can generate various solutions - exact or approximate. These are generated by replies to the request

Y or v?, ODE Option, TStepsize?

which should be answered by

c1, c2, δ ENTER

c1 should be the character y or the character v. If c1 = y, the height y is stored at each grid point If c2 = v the velocity is stored.

c2 should be x e rk2 or rk4. If c2 = x, the exact solution will be found. If c2 = e, Euler's method of solving the ordinary differential equations (3) will be used. c2 = rk2 or rk4 means that the second or fourth order Runge-Kutta method will be used in solving differential equations. These solution techniques are discussed in Section E. δ is the step size in t to be used in calculating solution. y or v will be calculated at t = $t_o, t_o + \delta, t_o + 2\delta$ etc. The final value of t will be t was entered as t_f if the reply to "T or Y range?" was entered as t but if y was specified for latter option, the final value of t is that value that makes y(t) = y_f for this particular method of solution. For these reasons, one sets up a grid in t which is uniform except for the last value. Some care should be taken in choosing δ. One wants, maybe, around 100 grid points and so should choose $\delta \simeq .01\ (t_f - t_o)$. This is impossible to calculate ahead of time if y_f and not t_f is specified. In this case, one may wish to do some preliminary calculations with various t_f's to see roughly how long it will take to get to y_f.

After calculating away (which may take a while) the program summarizes the current solutions and then patiently waits for a single character to be entered at the keyboard. Possibilities are:

a) Generate another solution - type M

b) List current solutions on screen, printer or a file - type L

c) Give up - type E

d) Restart - type R

e) Plot - type P

As in problem I 1, these characters must not be followed by ENTER We now describe the options R or P in more detail

After typing R, one is prompted again - if you type R yet again the program grudgingly deletes all its good work and we start again. If you type 0, this admits a user error and one is back at the previous command line If you type N, the previous solutions are not deleted but one is prompted for new parameters. This allows one to build up and later plot together, solutions with different values of a2 Thereby, one could address problem 5 1 on page 211 of Bennett 76

After typing P, one's actions are very similar to those already discussed in I.1. The program helps you choose the plot ranges by displaying the minimum and maximum values of time, height and velocity over all solutions Note that the plot can display velocities or heights versus time The choice is made by the user selecting which solutions to plot Each solution contains either velocity (v) **or** height (y). To generate both v and y for a particular set of parameters, one must treat this as two solutions generated as above by typing M.

The plot can display upto four solutions and these are selected by replies to the requests

0?

1?

2?

3?

where these are four curves produced in plot program. Reply with either a solution number or -1 (to ignore this curve)

Use the Zoom option to display better the small differences between the various solutions

E: Notes on Numerical Analysis: Integration of Ordinary Differentiation Equations.

The standard theory of the numerical solution of ordinary (i.e. one independent variable - in our case, time t) differential equations, refers to first order equations which can be written - as in Burden 76 -

$$dy/dt = f(t,y) \tag{1}$$

We convert the second order equation coming from Newton's laws into the two first order equations given in Section C, equation (3) The standard algorithms apply equally well to one or a system (i.e. more than one) of first order equations This theory is described fully in Chapter 5 of Burden 76 The algorithms called "e", "rk2" and "rk4" in ONEDIM1.BAS implement the fomalism given in algorithm 5.1, equation (5.3a) and algorithm 5.2 of Burden 76 respectively. The essential idea is already present in Euler's method which replaces the derivative in (1) by the forward difference to get:

$$(f(t+\delta) - f(t))/\delta = f(t,y(t)) \tag{2}$$

or

$$f(t+\delta) = f(t) + \delta f(t,y(t)) \qquad (3)$$

Clearly applying (3) recursively allows one to step t by δ at each stage and so integrate from t_0 to any final t There is nothing sacred about the exact step size δ, and the program uses equal steps δ upto the last stage which always uses a smaller step than δ. This smaller step value is trivial to calculate if one is integrating over a fixed t range, $t_0 \le t \le t_f$ but requires the techniques discussed in G if we have to obtain a particular y range, $y_0 \le y \le y_f$.

We know that the error in using (2) is of order δ (written O(δ)) and we have already discovered in Lesson I.1 that there are more accurate representations of the derivative than the forward difference (2) In fact, the central difference, defined in I 1, has an error of order δ^2 The mothod called rk2, or the modified Euler method, has an error of order δ^2 which it achieves by essentially using the central difference technique.

The Runge-Kutta label is applied to the class of methods that achieve higher order approximations to the (average) derivative between t and t + δ by calculating the functions at t values intermediate between the initial and final t value of each step. The method labelled rk4 is relatively simple to implement and has an error of O(δ^4).

Another class of methods - **none** of which are implemented in the example - obtain a higher order approximation by combining more than two grid point values e.g the values of y and dy/dt at t, t - δ, t - 2 δ... This class of method (multistep) is discussed in Section 5 6 of Burden 76 It has the disadvantage that it cannot be used to start the integration e g. when taking the initial step from t_0 to $t_0+\delta$, one does not have the necessary values corresponding to $t_0-\delta$, $t_0-2\delta$ .. for the multistep methods Typically one starts the multistep methods with a few Runge-Kutta steps to generate the initial few grid point values. The advantage of the multistep methods over the Runge-Kutta is that the latter "wastes" the intermediate calculation steps whereas the multistep obtains its accuracy by using values that have already been calculated.

Section J of this lesson discusses a sophisticated Runge-Kutta algorithm implemented in FORTRAN.

F: Notes on Numerical Analysis: Interpolation

Interpolation is used extensively in scientific calculations. There are two distinct classes of applications; first to the interpolating of experimental data and second to the interpolating of theoretical (or "exact") numbers. The two cases use different techniques because one must usually be concerned with the measurement errors in the data for the first case and so used method typified by the χ^2 fit. Assuming the rounding (and integration) errors are negligible, we can assume that the numbers coming from a theoretical calculation are exact.

Suppose we have a function f(t) and a collection of values (t_i, f_i) at N times t_0 ..$t_{(N-1)}$.We wish to find the value of f(t) at some general position t which is not in general equal to one of the t_i. In our case, t_i are the grid points used in the integration i.e. $t_0, t_1 = t_0 + \delta$, $t_2 = t_0 + 2\delta$,.. . As we explained, the set $\{t_i\}$ are in our application, essentially equispaced and although this could be useful in the computer program, this feature is not important for the general discussion of interpolation strategy. Note that the only reason we use interpolation at all is to speed up the calculation. We could, in fact, calculate v(t) or y(t) for any t by integrating the differential equation However, we choose to use the integrator to calculate v(t) and y(t) at a set of (closely - spaced) points and use interpolation to find the functions away from the grid points This technique also makes the program more modular Namely, interpolation can be implemented for any table (t_i, y_i) and does not need to be changed for tables that come from different sources

The simplest problem is linear interpolation Given (t_i, f_i) and (t_{i+1}, f_{i+1}) we write down the straight line (linear function) going through these two points in the (t,f) plane. This is

$$f_L(t) = \frac{f_{i+1}(t - t_i) + f_i(t_{i+1} - t)}{t_{i+1} - t_i} \qquad (1)$$

The strategy used in ONEDIM1 BAS, is to assume - as is automatic in our case - that the set $\{t_i\}$ are strictly increasing

$$t_0 < t_1 < t_2 .. < t_{N-1} \qquad (2)$$

Then given any value t, we search the list to find the pair t_i, t_{i+1} such that

$$t_i < t < t_{i+1} \qquad (3)$$

Having bracketed the target t, we apply the linear form (1) This technique, although simple, has proved to be generaly useful in a wide variety of problems In implementing it, one must decide what to do if $t < t_0$ or $t > t_{N-1}$ when extrapolation is necessary This is particularly unreliable and the user should not use this simple method if significant extrapolation will be necessary.

Lagrange's method generalizes (1) to find an n'th order polynomial that agrees with (n + 1) points of the set (t_i, f_i) Again it can be implemented by first searching for the n nearest grid points to the target t. In practice, high order polynomial interpolations are not used because they are potentially unreliable; physically this is because such polynomials have lots of wiggles and deviations of the fitted data from the polynomial can manifest itself as uncontrolled wiggles - especially near the end of the fitting region Lower order fits may not be very accurate but also they do not go wrong in ridiculous and unpredictable ways - a straight line doesn't wiggle! So usually one finds that the best approximation of a function over an interval is not a single very high order polynomial but a set of low order polynomials The cubic spline is a compromise that is often a good approach combining accuracy with reliability

G: Notes on Numerical Analysis: Solution of Equations in One Variable

This is discussed in Chapter 2 of Burden 76 and Section 21.6 of Roller 81. In general we wish to solve

$$f(t) = 0 \tag{1}$$

where in our case $f(t) = y(t) - y_f$. The case we have is particularly easy for two reasons

(a) Remember that we integrate at successive t's with interval δ until we find a value t_B with the corresponding $y(t_B) > y_f$. Choose the first t value that satisfies this so that $y(t_A = t_B - \delta) < y_f$. Thus we have obtained two values t_B and t_A such that

$f(t_B) < 0$

$f(t_A) > 0$

and so we know that the solution t_f satisfies $t_A < t_f < t_B$

This allows one to use the bisection method currently implemented in ONEDIM1 BAS

(b) We know the derivative f'(t) This makes Newton's method particularly easy to apply.

Bisection Method

In this, one successively halves the error in t_f by calculating the value of f(t) at the midpoint $t = 1/2(t_A + t_B)$. One replaces the range $t_A \to t_B$ by either $t_A \to 1/2(t_A + t_B)$ or $1/2(t_A + t_B) \to t_B$. The advantage of this method is that it is fool proof - one is bound to converge to a zero of f(t) Its disadvantage is that it is comparitively slow

Newton's Method

This is based on Taylor series expansion

$$f(t) = f(t_A) + f'(t_A)(t - t_A) \tag{2}$$

which gives an estimate

$$f(t_f) = 0$$

$$t_f = t_A - f(t_A)/f'(t_A) \tag{3}$$

Again we apply (3) iteratively. This converges faster because the error squares each time (being proportional to $(t_f - t_A)^2$) However, the faster convergence has a price Namely, you may get completely the wrong answer! This disaster will occur if $f'(t_A)$ changes sign between t_A and t_f As long as one is near enough the desired solution t_f (and the derivative is well behaved), these problems will not arise and the method will converge. In a proper implementation of Newton's method; one needs to check and see if it is converging. If it is, Newton's method will give the correct answer. If the method diverges, one should switch to a more conservative but slower method like the bisection technique. Although this detail will probably not be important in this course, it gives such attention to pathological cases that it marks the difference between an academic implementation of an algorithm and the practical version that can be used routinely in scientific work.

The reader is invited to improve this section by the construction of real examples with figures to illustrate graphically the main points (e.g cases which succeed/fail with Newton's method)

H: Suggested Problems

(a) As set up, ONEDIM1.BAS can (essentially) solve problems 1 and 2 on page 211 of Bennett 76.Work through these two problems and modify the program to address problems 3 -> 7.

(b) Read Section 5.6 of Eisberg 81. Use the program to reproduce fig 5-19 and 5-20. Address problems 5-43 to 5-48 of Eisberg 81.

(c) Examine the effectiveness of our ODE integrators by tackling problems 6.34 to 6.38 of Roller 81. Exercise set 5-2, 5-3, and 5-4 of Burden 76 also contain many such examples

(d) Change the problem to simulate one dimensional motion of a rocket as discussed on pages 186-190 of Eisberg 81. Address problems 5-49 to 5-53 of Eisberg 81. A similar discussion will be found in Section 9.6 and problems 9 34 to 9 38 of Roller 81.

(e) Run the program with $a_2 = 0$ Note that the Runge-Kutta technique gives exact answers at the grid points whereas Euler's method doesn't! Why is this? To show this effectively run with t range 0 to 1 and $\delta = 0.5$

(f) In (e), note that the display program joins grid point values with straight lines and so the Runge-Kutta **display** is not exact Improve the interpolation routine to correct this

(g) The velocity v(t) will eventually reach the asymptotic value $\sqrt{a_0 / a_2}$ Calculate this in the program and display this asymptotic line on plots of v versus time.

(h) Break the program into two parts - calculation and display - which communicate by writing a file (which contains VALT, VALY etc.). Run the calculation part with either the BASIC or C compiler What is the speed up compared to the interpreter? Use the 8087 - again what is the speed up

(i) Continuing (h), write the calculation program to use either display program built in to ONEDIM1 BAS or our standard Physics 20 plot package

(k) Improve the numerical integrator section to include options that allow other methods such as higher order Runge-Kutta or predictor corrector techniques discussed in Chapter 5 of Burden 76

(l) Improve the interpolation section to allow higher order approximations (see (f)) Investigate the accuracy of the interpolation as a function of the interpolation order. Is a higher order (Lagrange) interpolation always better than a lower order? Difference plots (See Section 1 1 problems) will be helpful here Take the cubic spline (p 107, Burden 76) as an example of a lower order interpolation

(m) Investigate other techniques for solving the equation f(t) = y_f (e.g. the Newton-Raphson method described in Section G is attractive as derivatives are known) needed when integrating to a definite height rather than a definite line

(n) Study for the built in integrators or new ones, both the error in a single step and the total error on completion of the integration. Characterize the success of the integration by two numbers, the average error and the maximum error (in y for $t_0 \le t \le t_f$). Find these as a function of δ and integration technique.

(o) Change the program to calculate y = y(t) for a particle falling from (x_0, y_0) to (x_f, y_f) on an arbitrary path y(x) with $y(x_0) = y_0$, $y(x_f) = y_f$. Compare transit times for various paths What is the minimum time. Make an arcade game where the player specifies y(x) interactively by, for instance, giving the instantaneous direction and distance to be travelled What is minimum time if one travels from (x_0, y_0) to (x_f, y_f) not by an arbitrary smooth curve but as in arcade game by a, possibly fixed, number N of straight line segments. Find the minimum time as a function of N.

(p) Improve the discussion in Sections E, F, G by producing appropriate figures (stored as tables on a disk file and plotted by our standard package) and illustrative examples.

(q) Is single presicion floating point sufficient..

I: Exact Solution of the Drag Problem

This is given on page 197 of Eisberg 81 for the special case when initial position and velocity are zero

Consider the differential equation

$$\dot{y} = g - \gamma y^2 \qquad (1)$$

This uses notation of Eisberg 81. Bennett 76 uses a_0 for g, a_2 for γ

Set

$$\tau = (\gamma g)^{1/2} t \qquad (2)$$

$$\xi = \gamma y \qquad (3)$$

Then (1) becomes (. now means d/d τ)

$$\dot{\xi} = 1 - \xi^2 \qquad (4)$$

which can be integrated once to give

$$\log\left[\frac{(1+\xi)}{(1-\xi)}\right] = 2(\tau + c) \qquad (5)$$

$$2c = \log[(1+\xi_o)/(1-\xi_o)] \qquad (6)$$

with $\xi = \xi_o$ at t = 0

Exponentiating both sides of (5) and manipulating gives

$$\frac{d\tau}{\tanh(\tau + c)} = d\xi \qquad (7)$$

which integrates to give

$$\xi - \xi_o = \log[\cosh(\tau + c)/\cosh c] \qquad (8)$$

with $\xi = \xi_o$ at t = o

We now have the final solution

$$\gamma(y - y_o) = \log[\cosh((\gamma g)^{1/2} t + c)/\cosh c] \qquad (9)$$

$$\gamma\dot{y} = (\gamma g)^{1/2}\tanh((\gamma g)^{1/2} t + c)$$

with $v_o = (\gamma/g)^{1/2}\, dy/dt\,|_o$

$2c = \log[(1 + v_o)/(1 - v_o)]$

and $y = y_o$ at t = 0.

(9) is valid as long as $v_o < 1$.

If $v_o > 1$, then

$$\gamma(y - y_o) = \log[\sinh((\gamma g)^{1/2} t + c)/\sinh c] \qquad (10)$$

$$\gamma\dot{y} = (\gamma g)^{1/2}\coth((\gamma g)^{1/2} t + c)$$

(9) and (10) are invalid if $v_o < 0$ as then (1) is invalid. One needs to integrate $\ddot{y} = g + \gamma\dot{y}^2$ when $\dot{y} < 0$. One can, of course, use similar techniques to integrate this case when v_o is < 0. We leave this to the reader.

J: A Sophisticated Runge-Kutta Algorithm

The IMSL corporation supplies a wide range of sophisticated scientific routines for a variety of computers. Currently they do not support the IBM PC due to the lack of DOUBLE PRECISION and COMPLEX statements in the FORTRAN compiler supplied with the PC. IMSL has generously allowed us to experiment with their software in this course.

The directory DVERK contains their Runge-Kutta routine and working test routines for two examples The algorithm used by DVERK is a more sophisticated version of that described in Section 5 5 of Burden 76 The method includes an error estimate for each step which allows one to reduce the step size if the estimated error is too large

We include the IMSL description of their routine

```
IMSL ROUTINE NAME   - DVERK

PURPOSE             - DIFFERENTIAL EQUATION SOLVER - RUNGE
                        KUTTA-VERNER FIFTH AND SIXTH ORDER METHOD

USAGE               - CALL DVERK (N,FCN,X,Y,XEND,TOL,IND,C,NW,W,IER)

ARGUMENTS    N      - NUMBER OF EQUATIONS. (INPUT)
             FCN    - NAME OF SUBROUTINE FOR EVALUATING FUNCTIONS.
                        (INPUT)
                       THE SUBROUTINE ITSELF MUST ALSO BE PROVIDED
                         BY THE USER AND IT SHOULD BE OF THE
                         FOLLOWING FORM
                           SUBROUTINE FCN(N,X,Y,YPRIME)
                             REAL Y(N),YPRIME(N)
                                  .
                                  .
                                  .
                       FCN SHOULD EVALUATE YPRIME(1),...,YPRIME(N)
                         GIVEN N,X, AND Y(1),...,Y(N).  YPRIME(I)
                         IS THE FIRST DERIVATIVE OF Y(I) WITH
                         RESPECT TO X.
                       FCN MUST APPEAR IN AN EXTERNAL STATEMENT IN
                         THE CALLING PROGRAM AND N,X,Y(1),...,Y(N)
                         MUST NOT BE ALTERED BY FCN.
             X      - INDEPENDENT VARIABLE. (INPUT AND OUTPUT)
                       ON INPUT, X SUPPLIES THE INITIAL VALUE.
                       ON OUTPUT, X IS REPLACED WITH XEND UNLESS
                         ERROR CONDITIONS ARISE.  SEE THE DES-
                         CRIPTION OF PARAMETER IND.
             Y      - DEPENDENT VARIABLES, VECTOR OF LENGTH N.
                        (INPUT AND OUTPUT)
                       ON INPUT, Y(1),...,Y(N) SUPPLY INITIAL
                         VALUES.
                       ON OUTPUT, Y(1),...,Y(N) ARE REPLACED WITH
                         AN APPROXIMATE SOLUTION AT XEND UNLESS
                         ERROR CONDITIONS ARISE.  SEE THE DES-
                         CRIPTION OF PARAMETER IND.
             XEND   - VALUE OF X AT WHICH SOLUTION IS DESIRED.
                        (INPUT)
                       XEND MAY BE LESS THAN THE INITIAL VALUE OF
                         X.
             TOL    - TOLERANCE FOR ERROR CONTROL. (INPUT)
                       THE SUBROUTINE ATTEMPTS TO CONTROL A NORM
                         OF THE LOCAL ERROR IN SUCH A WAY THAT THE
                         GLOBAL ERROR IS PROPORTIONAL TO TOL.
                         MAKING TOL SMALLER IMPROVES ACCURACY AND
                         MORE THAN ONE RUN, WITH DIFFERENT VALUES
                         OF TOL, CAN BE USED IN AN ATTEMPT TO
                         ESTIMATE THE GLOBAL ERROR.
                       IN THE DEFAULT CASE (IND=1), THE GLOBAL
                         ERROR IS
                             MAX(ABS(E(1)),...,ABS(E(N)))
                         WHERE E(K)=(Y(K)-YT(K))/MAX(1,ABS(Y(K)))
                         YT(K) IS THE TRUE SOLUTION, AND
```

```
                    Y(K) IS THE COMPUTED SOLUTION AT XEND,
                    FOR K=1,2,...,N.
                    OTHER ERROR CONTROL OPTIONS ARE AVAILABLE.
                    SEE THE DESCRIPTION OF PARAMETERS IND AND
                    C BELOW.
         IND    - INDICATOR. (INPUT AND OUTPUT)
                  ON INITIAL ENTRY IND MUST BE SET EQUAL TO
                  EITHER 1 OR 2.
                  IND = 1 CAUSES ALL DEFAULT OPTIONS TO BE
                    USED AND ELIMINATES THE NEED TO SET
                    SPECIFIC VALUES IN THE COMMUNICATIONS
                    VECTOR C.
                  IND = 2 ALLOWS OPTIONS TO BE SELECTED.  IN
                    THIS CASE, THE FIRST 9 COMPONENTS OF C
                    MUST BE INITIALIZED TO SELECT OPTIONS AS
                    DESCRIBED BELOW.
                THE SUBROUTINE WILL NORMALLY RETURN WITH
                  IND = 3, HAVING REPLACED THE INITIAL VALUES
                  OF X AND Y WITH, RESPECTIVELY, THE VALUE
                  XEND AND AN APPROXIMATION TO Y AT XEND.
                THE SUBROUTINE CAN BE CALLED REPEATEDLY WITH
                  NEW VALUES OF XEND WITHOUT CHANGING ANY
                  OF THE OTHER PARAMETERS.
                THREE ERROR RETURNS ARE ALSO POSSIBLE, IN
                  WHICH CASE X AND Y WILL BE THE MOST
                  RECENTLY ACCEPTED VALUES.
                  IND = -3 INDICATES THAT THE SUBROUTINE WAS
                    UNABLE TO SATISFY THE ERROR REQUIREMENT.
                    THIS MAY MEAN THAT TOL IS TOO SMALL.
                  IND = -2 INDICATES THAT THE VALUE OF HMIN
                    (MINIMUM STEP-SIZE) IS GREATER THAN HMAX
                    (MAXIMUM STEP-SIZE), WHICH PROBABLY MEANS
                    THAT THE REQUESTED TOL (WHICH IS USED IN
                    THE CALCULATION OF HMIN) IS TOO SMALL.
                  IND = -1 INDICATES THAT THE ALLOWED MAXIMUM
                    NUMBER OF FCN EVALUATIONS HAS BEEN
                    EXCEEDED.  THIS CAN ONLY OCCUR IF OPTION
                    C(7), AS DESCRIBED BELOW, HAS BEEN USED.
         C      - COMMUNICATIONS VECTOR OF LENGTH 24. (INPUT IF
                  IND.NE.1, AND OUTPUT).
                  C IS USED TO SELECT OPTIONS AND TO RETAIN
                    INFORMATION BETWEEN CALLS.  THE USER NEED
                    NOT BE CONCERNED WITH THE FOLLOWING
                    DESCRIPTION OF THE ELEMENTS OF C WHEN
                    DEFAULT OPTIONS ARE USED (IND=1).
                    HOWEVER, WHEN IT IS DESIRED TO USE IND=2
                    AND SELECT OPTIONS, A BASIC UNDERSTANDING
                    OF DVERK IS REQUIRED.  THE FOLLOWING
                    PARAGRAPH DESCRIBES, BRIEFLY, THE BASIC
                    TERMS.  FOR MORE DETAILS, SEE THE DOCUMENT
                    REFERENCE.
                    DVERK ADVANCES THE INDEPENDENT VARIABLE
                    X ONE STEP AT A TIME UNTIL XEND IS
                    REACHED.  THE SOLUTION IS COMPUTED AT
                    XTRIAL = X+HTRIAL ALONG WITH AN ERROR
                    ESTIMATE EST.  IF EST IS LESS THAN OR
                    EQUAL TO TOL (SUCCESSFUL STEP), THE STEP
                    IS ACCEPTED AND X IS ADVANCED TO XTRIAL.
```

```
                       IF EST IS GREATER THAN TOL (FAILURE)
                       HTRIAL IS ADJUSTED AND THE SOLUTION IS
                       RECOMPUTED.  HMAG = ABS(HTRIAL) IS NEVER
                       ALLOWED TO EXCEED HMAX NOR IS IT ALLOWED
                       TO BECOME SMALLER THAN HMIN.  THE FIRST
                       TRIAL STEP IS HSTART.  DURING THE
                       COMPUTATION, A COUNTER (C(23)) IS
                       INCREMENTED EACH TIME A TRIAL STEP FAILS
                       TO PROVIDE A SOLUTION SATISFYING THE ERROR
                       TOLERANCE.  ANOTHER COUNTER (C(22)) IS
                       USED TO RECORD THE NUMBER OF SUCCESSFUL
                       STEPS.  AFTER A SUCCESSFUL STEP, C(23) IS
                       SET TO ZERO.
                  OPTIONS.  IF THE SUBROUTINE IS ENTERED WITH
                    IND=2, THE FIRST 9 COMPONENTS OF THE
                    COMMUNICATIONS VECTOR MUST BE INITIALIZED
                    BY THE USER.  NORMALLY THIS IS DONE BY
                    FIRST SETTING THEM ALL TO ZERO, AND THEN
                    THOSE CORRESPONDING TO PARTICULAR OPTIONS
                    ARE MADE NON-ZERO.
         C(1)     - ERROR CONTROL INDICATOR.
                       THE SUBROUTINE ATTEMPTS TO CONTROL A NORM
                       OF THE LOCAL ERROR IN SUCH A WAY THAT THE
                       GLOBAL ERROR IS PROPORTIONAL TO TOL.
                       THE DEFINITION OF GLOBAL ERROR FOR THE
                       DEFAULT CASE (IND=1) IS GIVEN IN THE
                       DESCRIPTION OF PARAMETER TOL.  THE DEFAULT
                       WEIGHTS ARE 1/MAX(1,ABS(Y(K))).  WHEN IND=2
                       IS USED, THE WEIGHTS ARE DETERMINED
                       ACCORDING TO THE VALUE OF C(1).
                       IF C(1)=1 THE WEIGHTS ARE 1
                                 (ABSOLUTE ERROR CONTROL)
                       IF C(1)=2 THE WEIGHTS ARE 1/ABS(Y(K))
                                 FOR K=1,2,...,N.
                                 (RELATIVE ERROR CONTROL)
                       IF C(1)=3 THE WEIGHTS ARE
                                 1/MAX(ABS(C(2)),ABS(Y(K)))
                                 FOR K=1,2,...,N.
                                 (RELATIVE ERROR CONTROL, UNLESS
                                 ABS(Y(K)) IS LESS THAN THE FLOOR
                                 VALUE,ABS(C(2)))
                       IF C(1)=4 THE WEIGHTS ARE
                                 1/MAX(ABS(C(K+30)),ABS(Y(K)))
                                 FOR K=1,2,...,N.
                                 (HERE INDIVIDUAL FLOOR VALUES
                                 ARE USED)
                                 IN THIS CASE,  THE DIMENSION OF C
                                 MUST BE GREATER THAN OR EQUAL TO
                                 N+30 AND C(31), C(32),...,C(N+30)
                                 MUST BE INITIALIZED BY THE USER.
                       IF C(1)=5 THE WEIGHTS ARE 1/ABS(C(K+30))
                                 FOR K=1,2,...,N.
                                 IN THIS CASE,  THE DIMENSION OF C
                                 MUST BE GREATER THAN OR EQUAL TO
                                 N+30 AND C(31), C(32),...,C(N+30)
                                 MUST BE INITIALIZED BY THE USER.
```

```
                FOR ALL OTHER VALUES OF C(1), INCLUDING
                  C(1)=0 THE DEFAULT VALUES OF
                         THE WEIGHTS ARE TAKEN TO BE
                         1/MAX(1,ABS(Y(K)))
                            FOR K=1,2,...,N.
     C(2)   - FLOOR VALUE.  USED WHEN THE INDICATOR C(1)
               HAS THE VALUE 3.
     C(3)   - HMIN SPECIFICATION.  IF NOT ZERO, THE SUB-
               ROUTINE CHOOSES HMIN TO BE ABS(C(3)).
               OTHERWISE IT USES THE DEFAULT VALUE
               10*MAX(DWARF,RREB*MAX(NORM(Y)/TOL,ABS(X)))
               WHERE DWARF IS A VERY SMALL POSITIVE MACHINE
               NUMBER AND RREB IS THE RELATIVE ROUNDOFF
               ERROR BOUND.
     C(4)   - HSTART SPECIFICATION.  IF NOT ZERO, THE SUB-
               ROUTINE WILL USE AN INITIAL HMAG EQUAL TO
               ABS(C(4)), EXCEPT OF COURSE FOR THE RE-
               STRICTIONS IMPOSED BY HMIN AND HMAX.
               OTHERWISE IT USES THE DEFAULT VALUE
                 HMAX*(TOL)**(1/6).
     C(5)   - SCALE SPECIFICATION.  THIS IS INTENDED TO BE
               A MEASURE OF THE SCALE OF THE PROBLEM.
               LARGER VALUES OF SCALE TEND TO MAKE THE
               METHOD MORE RELIABLE, FIRST BY POSSIBLY RE-
               STRICTING HMAX (AS DESCRIBED BELOW) AND
               SECOND, BY TIGHTENING THE ACCEPTANCE
               REQUIREMENT.  IF C(5) IS ZERO, A DEFAULT
               VALUE OF 1 IS USED. FOR LINEAR HOMOGENEOUS
               PROBLEMS WITH CONSTANT COEFFICIENTS, AN
               APPROPRIATE VALUE FOR SCALE IS A NORM OF
               THE ASSOCIATED MATRIX. FOR OTHER PROBLEMS,
               AN APPROXIMATION TO AN AVERAGE VALUE OF A
               NORM OF THE JACOBIAN ALONG THE TRAJEC-
               TORY MAY BE APPROPRIATE.
     C(6)   - HMAX SPECIFICATION.  FOUR CASES ARE POSSIBLE,
               IF C(6).NE.0 AND C(5).NE.0, HMAX IS TAKEN
                 TO BE MIN(ABS(C(6)),2/ABS(C(5))).
               IF C(6).NE.0 AND C(5).EQ.0, HMAX IS TAKEN
                 TO BE ABS(C(6)).
               IF C(6).EQ.0 AND C(5).NE.0, HMAX IS TAKEN
                 TO BE 2/ABS(C(5)).
               IF C(6).EQ.0 AND C(5).EQ.0, HMAX IS GIVEN
                 A DEFAULT VALUE OF 2.
     C(7)   - MAXIMUM NUMBER OF FUNCTION EVALUATIONS.  IF
               NOT ZERO, AN ERROR RETURN WITH IND = -1
               WILL BE CAUSED WHEN THE NUMBER OF FUNCTION
               EVALUATIONS EXCEEDS ABS(C(7)).
     C(8)   - INTERRUPT NUMBER 1 . IF NOT ZERO, THE SUB-
               ROUTINE WILL INTERRUPT THE CALCULATIONS
               AFTER IT HAS CHOSEN ITS PRELIMINARY VALUE
               OF HMAG, AND JUST BEFORE CHOOSING HTRIAL
               AND XTRIAL IN PREPARATION FOR TAKING A STEP
               (HTRIAL MAY DIFFER FROM HMAG IN SIGN, AND
               MAY REQUIRE ADJUSTMENT IF XEND IS NEAR).
               THE SUBROUTINE RETURNS WITH IND = 4, AND
               WILL RESUME CALCULATION AT THE POINT OF
               INTERRUPTION IF  RE-ENTERED WITH IND = 4.
```

```
                C(9)   - INTERRUPT NUMBER 2.  IF NOT ZERO, THE SUB-
                           ROUTINE WILL INTERRUPT THE CALCULATIONS
                           IMMEDIATELY AFTER IT HAS DECIDED WHETHER OR
                           NOT TO ACCEPT THE RESULT OF THE MOST RECENT
                           TRIAL STEP, WITH IND = 5 IF IT PLANS TO
                           ACCEPT, OR IND = 6 IF IT PLANS TO REJECT.
                           Y(*) IS THE PREVIOUSLY ACCEPTED RESULT,
                           WHILE W(*,9) IS THE NEWLY COMPUTED TRIAL
                           VALUE, AND W(*,2) IS THE UNWEIGHTED ERROR
                           ESTIMATE VECTOR. THE SUBROUTINE WILL RESUME
                           CALCULATIONS AT THE POINT OF INTERRUPTION
                           ON RE-ENTRY WITH IND = 5 OR 6.
                           IND MAY BE CHANGED BY THE USER IN ORDER TO
                           FORCE ACCEPTANCE OF A STEP (BY CHANGING IND
                           FROM 6 TO 5) THAT WOULD OTHERWISE BE
                           REJECTED, OR VICE VERSA.
              NW       - ROW DIMENSION OF THE MATRIX W EXACTLY AS
                           SPECIFIED IN THE DIMENSION STATEMENT
                           IN THE CALLING PROGRAM. (INPUT)
                           NW MUST BE GREATER THAN OR EQUAL TO N.
              W        - WORKSPACE MATRIX.
                           THE FIRST DIMENSION OF W MUST BE NW AND THE
                           SECOND MUST BE GREATER THAN OR EQUAL TO 9.
                           W MUST REMAIN UNCHANGED BETWEEN SUCCESSIVE
                           CALLS DURING INTEGRATION.
              IER      - ERROR PARAMETER. (OUTPUT)
                         TERMINAL ERROR
                           IER = 129, NW IS LESS THAN N OR TOL IS LESS
                             THAN OR EQUAL TO ZERO.
                           IER = 130, IND IS NOT IN THE RANGE 1 TO 6.
                           IER = 131, XEND HAS NOT BEEN CHANGED FROM
                             PREVIOUS CALL OR X IS NOT SET TO
                             THE PREVIOUS XEND VALUE.
                           IER = 132, THE RELATIVE ERROR CONTROL
                             OPTION (C(1)=2) WAS SELECTED AND
                             ONE OF THE SOLUTION COMPONENTS
                             IS ZERO.

PRECISION/HARDWARE   - SINGLE AND DOUBLE/H32
                     - SINGLE/H36,H48,H60

REQD. IMSL ROUTINES - UERTST,UGETIO

NOTATION             - INFORMATION ON SPECIAL NOTATION AND
                         CONVENTIONS IS AVAILABLE IN THE MANUAL
                         INTRODUCTION OR THROUGH IMSL ROUTINE UHELP

REMARKS  1.  IN A TYPICAL SITUATION, DVERK IS CALLED
             REPEATEDLY WITH A SEQUENCE OF VALUES FOR XEND.
             AFTER EACH SUCH CALL, THE USER SHOULD INTERROGATE
             IND AND IER. ERROR CONDITIONS ARE SIGNALED WHEN
             IND IS LESS THAN ZERO AND/OR IER IS GREATER THAN
             ZERO. CORRECTIVE ACTION (SUCH AS CHANGING CERTAIN
             PARAMETER VALUES) MUST BE TAKEN PRIOR TO RE-ENTRY.
         2.  WHEN ERROR CONDITIONS ARISE, IT IS OFTEN HELPFUL
             TO EXAMINE COMPONENTS OF THE COMMUNICATIONS VECTOR
             C. A SUMMARY FOLLOWS-
```

```
PRESCRIBED AT THE OPTION OF THE USER

  C(1) ERROR CONTROL INDICATOR
  C(2) FLOOR VALUE
  C(3) HMIN SPECIFICATION
  C(4) HSTART SPECIFICATION
  C(5) SCALE SPECIFICATION
  C(6) HMAX SPECIFICATION
  C(7) MAXIMUM NUMBER OF FCN EVALUATIONS
  C(8) INTERRUPT NUMBER 1
  C(9) INTERRUPT NUMBER 2

DETERMINED BY THE PROGRAM

  C(10) RREB (RELATIVE ROUNDOFF ERROR BOUND)
  C(11) DWARF (VERY SMALL MACHINE NUMBER)
  C(12) WEIGHTED NORM OF Y
  C(13) HMIN
  C(14) HMAG
  C(15) SCALE
  C(16) HMAX
  C(17) XTRIAL
  C(18) HTRIAL
  C(19) EST
  C(20) PREVIOUS XEND
  C(21) FLAG FOR XEND
  C(22) NUMBER OF SUCCESSFUL STEPS
  C(23) NUMBER OF SUCCESSIVE FAILURES
  C(24) NUMBER OF FCN EVALUATIONS

IF C(1) = 4 OR 5, C(31),C(32),...,C(N+30) ARE FLOOR
VALUES.
```

3. PARAMETER NW GIVES THE ROW DIMENSION OF W EXACTLY AS IT APPEARS IN THE DIMENSION STATEMENT IN THE CALLING PROGRAM. IF ONLY ONE SYSTEM OF EQUATIONS IS BEING SOLVED, NW NORMALLY WILL HAVE THE SAME VALUE AS N. HOWEVER, IF MORE THAN ONE SYSTEM IS BEING HANDLED, AND THEY ARE TO USE A COMMON WORKSPACE, W, ONE AFTER THE OTHER, THE VALUE OF NW (AND HENCE, THE ROW DIMENSION OF W IN THE CALLING PROGRAM) MUST BE AS LARGE AS THE MAXIMUM VALUE OF THE INDIVIDUAL N VALUES.

Algorithm

DVERK finds approximations to the solution of a system of first order ordinary differential equations of the form y'=f(x,y) with initial conditions. It is designed to be easy to use. By setting parameter IND to 1, the user need only provide parameters to describe the problem; everything else is done automatically by the subroutine. Alternatively, the user may set IND to 2 and then select any one of several options, including different kinds of error control, restrictions on step sizes, and interrupts which permit the user to examine the state of the calculations (and perhaps make modifications) during intermediate stages. DVERK attempts to keep the global error proportional to a tolerance specified by the user. The proportionality depends on the kind of error control that is used as well as the differential equation and the range of integration.

DVERK is efficient for non-stiff systems where derivative evaluations are not expensive and where solutions are not required at a large number of finely spaced points (as might be the case for example with graphical output). See the Chapter D prelude for general guidelines.

The subroutine is based on a code designed by T. E. Hull, W. H. Enright, and K. R. Jackson. It uses Runge-Kutta formulas of orders 5 and 6 that were developed by J. H. Verner.

See references:

1. T. E. Hull, W. H. Enright, and K. R. Jackson, "User's Guide for DVERK - A Subroutine for Solving Non-Stiff ODE's", TR No. 100, Department of Computer Science, University of Toronto, October, 1976.

2. K. R. Jackson, W. H. Enright, and T. E. Hull, "A Theoretical Criterion for Comparing Runge-Kutta Formulas TR101", January, 1977.

Example 1

This example illustrates the basic usage (all default options) of DVERK. A table of solution values for $x = 1.0, 2.0, \ldots, 10.0$ is obtained for the predator-prey problem:

$$y_1' = 2y_1(1-y_2) \qquad y_1 = 1$$
$$\qquad\qquad\qquad\qquad\qquad\qquad \text{at } x = 0$$
$$y_2' = y_2(y_1-1) \qquad y_2 = 3$$

```
      INTEGER   N,IND,NW,IER,K
      REAL      Y(2),C(24),W(2,9),X,TOL,XEND
      EXTERNAL  FCN1
      NW    = 2
      N     = 2
      X     = 0.0
      Y(1)  = 1.0
      Y(2)  = 3.0
      TOL   = .0001
      IND   = 1
      DO 10 K=1,10
        XEND=FLOAT(K)
        CALL DVERK(N,FCN1,X,Y,XEND,TOL,IND,C,NW,W,IER)
        IF(IND.LT.0.OR.IER.GT.0) GO TO 20
C                    Y(1) and Y(2) are current solution values at X.
C                    Insert write statement here.
   10 CONTINUE
      STOP
   20 CONTINUE
```

```
C                     Handle IND.LT.0 or IER.GT.0
C                     Items that may help diagnose the problem should be
C                     output here.
C                     IND,TOL,N,W,Y(1),...,Y(N),XEND, and C(1),...,C(24).
      STOP
      END
      SUBROUTINE      FCN1(N,X,Y,YPRIME)
      INTEGER         N
      REAL            Y(N),YPRIME(N),X
      YPRIME(1) = 2.0*Y(1)*(1.0-Y(2))
      YPRIME(2) = Y(2)*(Y(1)-1.0)
      RETURN
      END
```

Output:

```
      IER = 0
                X              Y(1)           Y(2)
                1.             0.08           1.46
                2.             0.09           0.58
                3.             0.29           0.25
                4.             1.45           0.19
                5.             4.05           1.44
                6.             0.18           2.26
                7.             0.07           0.91
                8.             0.15           0.37
                9.             0.65           0.19
               10.             3.15           0.35
```

Example 2

This example shows how IND = 2 is used to select specific options, while using default values for others. The differential equation

$$y' = y \quad , \quad y = 1 \text{ at } x = 0,$$

is solved for x = .1,.2,...,1.0, using the absolute error control option (C(1)=1).

```
      INTEGER   N,IND,NW,IER,I,K
      REAL      Y(1),C(24),W(1,9),X,TOL,XEND
      EXTERNAL  FCN2
      NW    = 1
      N     = 1
      X     = 0.0
      Y(1)  = 1.0
      TOL   = 0.0005
      IND   = 2
C                     Select all default options, first
      DO 5 I=1,9
    5 C(I) = 0.0
C                      Then specify C(1)=1.0 to select the absolute error
C                      control option.
      C(1) = 1.0
      DO 10 K=1,10
        XEND = FLOAT(K)*0.1
        CALL DVERK(N,FCN2,X,Y,XEND,TOL,IND,C,NW,W,IER)
        IF(IND.LT.0.OR.IER.GT.0) GO TO 20
```

```
C                     Y(1) is the current solution value at X.  Insert write
C                     statement here.
 10 CONTINUE
    STOP
 20 CONTINUE
C                     Handle IND.LT.0 or IER.GT.0
C                     Items that may help diagnose the problem should be
C                     output here.
C                     IND,TOL,N,X,Y(1),...,Y(N),XEND, and C(1),...,C(24).
      STOP
      END
      SUBROUTINE      FCN2(N,X,Y,YPRIME)
      INTEGER         N
      REAL            Y(N),YPRIME(N),X
      YPRIME(1) = Y(1)
      RETURN
      END
```

Output:

```
IER = 0
          X          Y(1)
         .1          1.105
         .2          1.221
         .3          1.350
         .4          1.492
         .5          1.649
         .6          1.822
         .7          2.014
         .8          2.226
         .9          2.460
        1.0          2.718
```

K ONEDIM1 PROGRAM STRUCTURE

(1) LINES 100-110	Defines functions used by exact solution for fall under drag
(2) Lines 140-200	Defines arrays used by plot part of program
(3) Lines 230-290	Defines arrays used by specific (non-plot) part of program
(4) Lines 320-790	Reads in data
(5) Lines 820-1010	Calls subroutine (10) to calculate requested solution and stores results in the arrays VALT and VALY We have NTRY% solutions and for $0 \le i \le$ NTRY% - 1 the t values are stored in VALT (i1) to VALT (i1 + n - 1) where i1 = BEG%(i) and n = NUMENT%(i) VALY(i1) to VALY(i1 + n - 1) holds y values (KEEPYV$(i) = y) or v (velocity) values (KEEPYV$(i) = v)
(6) Lines 1040-1090	Summarizes current solutions
(7) Lines 1120-1320	Requests next action and sets it up
(8) Lines 1370-1680	Writes current solutions on a file Note code will only write as single table solutions which have the same set of t values
(9) Lines 1730-3630	Produces plots on colour monitor This uses subroutine (12) to interpolate in the table VALT, VALY produced in step (5)

(10) Lines 3820-4590 Is code that controls calculations It

a) Calls basic ODE integrators (11) to advance **solution** one step in t This control section loops over t steps.

b) Calculates exact solution if known If not known, an "exact" solution request is replaced by an ODE call (11) with the "rk4" (4th order Runge-Kutta) option.

c) If integrating over a specified y range, the control section calls the bisection algorithm (13) to converge on the desired y value

(11) Lines 4940-5390 Is a subroutine to integrate (a set of) ODE's **one** step in t according to "e", "rk2" or "rk4" algorithm. See Section E.

(12) Lines 5490-5690 Is a subroutine to interpolate in a table using algorithm described in Section F This subroutine is called by plot part of program

(13) Lines 5870-5950 Contains a subroutine implementing bisection algorithm described in Section G

Note that each subroutine is preceded by comments describing function of code and meaning of input and output variables

ONEDSM1.BAS

```
10 REM The numerical solution of ordinary differential equations
20 REM Units are MKS
30 REM
40 REM ------------------------------------------------------------
50 REM BEGIN setup section
60 REM
70 CLEAR
80 REM
90 REM Some useful Functions
100 DEF FNCOSH(X)=.5*(EXP(X)+EXP(-X))
110 DEF FNSINH(X)=.5*(EXP(X)-EXP(-X))
120 REM
130 REM Arrays for plot labels
140 DIM YLABEL(10)
150 DIM TLABEL(10)
160 DIM MESS0%(322)
170 DIM MESS1%(322)
180 DIM MESS2%(322)
190 DIM MESS3%(162):DIM MESS4%(162)
200 DIM MESS5%(162):DIM MESS6%(162)
210 REM
220 REM Arrays for problem
230 DIM INTOPT%(20),DEL(20),BEG%(20),NUMENT%(20),KEEPYV$(20),KEEPA2(20)
240 DIM VALT(2000),VALY(2000)
250 DIM TEMPT(400),TEMPY(400),TEMPV(400)
260 DIM NMDIFF$(3)
270 DATA "Exact","Euler","Runge Kutta 2nd order","Runge Kutta 4th order"
280 FOR I%=0 TO 3:READ NMDIFF$(I%):NEXT I%
290 DIM YEOU(1),DYEOU(1)
300 REM
310 REM ------------------------------------------------------------
320 REM BEGIN input section
330 REM
340 SCREEN 0,0:WIDTH 40
350 CLS
360 NTRY%=-1
370 USE%=0
380 MINT=1E+10:MAXT=-1E+10
390 MINY=1E+10:MAXY=-1E+10
400 MINV=1E+10:MAXV=-1E+10
410 KEY OFF
420 LOCATE 25,1
430 PRINT "M more L lists P plots R restarts E ends"
440 LOCATE 1,1
450 REM
460 REM Define problem with data from keyboard
470 A0DRAG=9.797
480 PRINT "Use units of Metres and Seconds"
490 INPUT "Drag Coefficient a2";A2DRAG
500 IF ABS(A2DRAG)<1E-09 THEN NODRAG%=1 ELSE NODRAG%=0
510 INPUT "Initial T,Y,dY/dT";TPHY1,YPHY1,YDERV1
520 INPUT "T or Y range";TYOPT$
530 IF TYOPT$="T" THEN TYOPT$="t"
540 IF TYOPT$<>"t" THEN TYOPT$="y"
550 IF TYOPT$="y" THEN GOTO 570
560 INPUT "Final T value";TPHY2:GOTO 580
570 INPUT "Final Y value";YPHY2
580 INPUT "Y or V?, ODE Option, Tstepsize";YVOPT$,OPT$,DELTA
590 REM
600 REM interpret and check option
610 IF YVOPT$="Y" THEN YVOPT$="y"
620 IF YVOPT$<>"y" THEN YVOPT$="v"
630 IPD%=-1
```

```
640 IF OPT$="x" OR OPT$="X" THEN IDO%=0
650 IF OPT$="E" OR OPT$="e" THEN IDO%=1
660 IF OPT$="rk2" OR OPT$="RK2" THEN IDO%=2
670 IF OPT$="rk4" OR OPT$="RK4" THEN IDO%=3
680 IF IDO% >-1 THEN GOTO 710
690 PRINT "Options are e x rk2 or rk4. Please try again"
700 GOTO 580
710 NTRY%=NTRY%+1
720 IF(NTRY%<21) GOTO 760
730 PRINT "Too many options. Either plot list or restart"
740 NTRY%=NTRY%-1
750 GOTO 1120
760 INTOPT%(NTRY%)=IDO%
770 DEL(NTRY%)=DELTA
780 KEEPYV$(NTRY%)=YVOPT$
790 KEEPA2(NTRY%)=A2DRAG
800 REM
810 REM Set up tables of t values and y values
820 GOSUB 3650
830 REM
840 REM Save calculated values
850 BEG%(NTRY%)=USE%
860 NUMENT%(NTRY%)=NSTEP%
870 FOR J%=1 TO NSTEP%
880 IF USE%  =2000 THEN GOTO 920
890 PRINT "Too many values(2000 is limit) "
900 USE%=BEG%(NTRY%)
910 GOTO 740
920 VALT(USE%)=TEMPT(J%)
930 IF MINT   TEMPT(J%) THEN MINT=TEMPT(J%)
940 IF MAXT   TEMPT(J%) THEN MAXT=TEMPT(J%)
950 IF YVOPT$="y" THEN VALY(USE%)=TEMPY(J%) ELSE VALY(USE%)=TEMPV(J%)
960 IF MINY   TEMPY(J%) THEN MINY=TEMPY(J%)
970 IF MAXY   TEMPY(J%) THEN MAXY=TEMPY(J%)
980 IF MINV   TEMPV(J%) THEN MINV=TEMPV(J%)
990 IF MAXV   TEMPV(J%) THEN MAXV=TEMPV(J%)
1000 USE%=USE%+1
1010 NEXT J%
1020 REM
1030 REM Summarize entries
1040 IF TYOPT$="t" THEN PRINT " T range:" TPHY1;TPHY2:PRINT "Initial y,dy/dt:" Y
PHY1;YDERV1
1050 IF TYOPT$="y" THEN PRINT " Y range:" YPHY1;YPHY2:PRINT "Initial t,dy/dt:" T
PHY1;YDERV1
1060 FOR I%=0 TO NTRY%
1070 PRINT "No: " I% " Option: " NMDIFF$(INTOPT%(I%)) " Delta: " DEL(I%)
1080 PRINT "No: " I% " Y or V option:" KEEPYV$(I%) " A2drag:" KEEPA2(I%)
1090 NEXT I%
1100 REM
1110 REM READ COMMAND FROM KEYBOARD
1120 K$=INKEY$
1130 IF K$="" THEN GOTO 1120
1140 IF K$="M" OR K$="m" THEN GOTO 580
1150 IF K$="P" OR K$="p" THEN GOTO 1730
1160 IF K$="L" OR K$="l" THEN GOTO 1370
1170 IF K$="R" OR K$="r" THEN GOTO 1200
1180 IF K$="E" OR K$="e" THEN GOTO 3340
1190 GOTO 1120
1200 PRINT " Total restart(r) New(n) Old(o) parameters"
1210 K$=INKEY$
1220 IF K$="" THEN GOTO 1210
1230 IF K$="N" OR K$="n" THEN GOTO 470
1240 IF K$="R" OR K$="r" THEN GOTO 350
1250 IF K$="O" OR K$="o" THEN GOTO 1290
1260 GOTO 1210
1270 REM
```

```
1280 REM Restart to add integration options
1290 SCREEN 0,0:CLS:LOCATE 25,1
1300 PRINT "M more L lists P plots R restarts E ends"
1310 LOCATE 1,1
1320 GOTO 1040
1330 REM
1340 REM -----------------------------------------------------------
1350 REM BEGIN output section: first on a file, then on screen
1360 REM Produce output in form for plot package
1370 NOW%=-1
1380 PRINT "Use scrn: for screen, lpt1: for printer"
1390 PRINT "Or any disk file name(s)"
1400 INPUT "file name"; FILE$
1410 OPEN FILE$ FOR OUTPUT AS #1
1420 REM
1430 REM discover which entries to output together
1440 NOW%=NOW%+1
1450 IF TYOPT$="y" THEN GOTO 1500
1460 FOR I%=NOW% TO NTRY%
1470 IF DEL(I%) <> DEL(NOW%) THEN GOTO 1510
1480 NEXT I%
1490 NEND%=NTRY%:GOTO 1520
1500 NEND%=NOW%:GOTO 1520
1510 NEND%=I%-1
1520 PRINT FILE$ " Contains entries " NOW% " to " NEND%
1530 REM
1540 REM List entries now% to nend% with same t labels
1550 FOR J%=1 TO NUMENT%(NOW%)
1560 KNOW%=BEG%(NOW%)+J%-1
1570 PRINT #1,USING "#.###      : ";VALT(KNOW%);
1580 FOR I%=NOW% TO NEND%
1590 K%=BEG%(I%)+J%-1
1600 PRINT #1,USING "#.###      , ";VALY(K%);
1610 NEXT I%
1620 PRINT #1,
1630 NEXT J%
1640 NOW%=NEND%
1650 IF NOW%-NTRY% THEN PRINT "End Listing":CLOSE #1:GOTO 1120
1660 INPUT "Next File Name";NFILE$
1670 IF FILE$=NFILE$ THEN GOTO 1440
1680 CLOSE #1:FILE$=NFILE$:GOTO 1410
1690 REM
1700 REM -----------------------------------------------------------
1710 REM BEGIN section to produce screen plots
1720 REM
1730 PRINT "Note that Plot uses Linear Interpolation"
1740 PRINT "Even though some ODE solvers are higher order"
1750 PRINT "T range stored",MINT,MAXT
1760 PRINT "Y range stored",MINY,MAXY
1770 PRINT "V range stored",MINV,MAXV
1780 INPUT "Plot T range";T1,T2
1790 INPUT "Plot Y range";Y1,Y2
1800 PRINT "You can select four trials (negative numbers ignored)"
1810 INPUT "0";ITRY0%:INPUT "1";ITRY1%:INPUT "2";ITRY2%:INPUT "3";ITRY3%
1820 REM
1830 REM Find range of plots
1840 TMIN=T1
1850 IF T1 T2 THEN  TMIN=T2
1860 TMAX=T2
1870 IF T2 T1  THEN TMAX=T1
1880 YMIN=Y1
1890 FULL%=0
1900 IF Y1 Y2 THEN YMIN=Y2
1910 YMAX=Y2
1920 IF Y2 Y1  THEN YMAX=Y1
1930 REM
```

```
1940 REM Set text strings holding labels for later use
1950 CLS
1960 SCREEN 1,0
1970 LOCATE 1,1
1980 IF KEEPYV$(ITRYO%)="y" THEN PRINT "y" ELSE PRINT "v"
1990 GET (0,0)-(7,7),YLABEL
2000 LOCATE 1,1
2010 PRINT "t"
2020 GET (0,0)-(7,7),TLABEL
2030 LOCATE 1,1
2040 PRINT "Cyan-0 White-1 Pink dot-2 dash-3"
2050 GET (0,0)-(319,7),MESS0%
2060 CLS:LOCATE 1,1
2070 PRINT "F Flips Full<->Half Size"
2080 GET (0,0)-(319,7),MESS1%
2090 CLS:LOCATE 1,1
2100 PRINT "E ends R restarts Z zooms P plots"
2110 GET (0,0)-(319,7),MESS2%
2120 CLS
2130 DT=.101*ABS(T2-T1)
2140 DY=.101*ABS(Y2-Y1)
2150 TBEG=TMIN-DT:TEND=TMAX+DT:YBEG=YMIN-DY:YEND=YMAX+DY
2160 TBEGO=TBEG:TENDO=TEND:YBEGO=YBEG:YENDO=YEND
2170 USEVIEW%=1
2180 REM
2190 REM Start Plot by deciding on graph axis limits
2200 IF USEVIEW%=1 THEN VIEW (0,0)-(158,190)
2210 IF USEVIEW%=2 THEN VIEW(160,0)-(318,190)
2220 IF USEVIEW%=3 THEN VIEW (0,0)-(319,190)
2230 WINDOW
2240 CLS
2250 DT=.1*ABS(T2-T1)
2260 DY=.1*ABS(Y2-Y1)
2270 TONE=TEND:TTWO=TBEG
2280 FOR T=TMIN TO TMAX+.001 STEP DT
2290 IF (T TBEG+DT) OR (T TEND-DT) THEN GOTO 2320
2300 IF T TONE THEN TONE=T
2310 IF T TTWO THEN TTWO=T
2320 NEXT T
2330 IF TONE  = TTWO THEN DT=.2*DT:GOTO 2270
2340 YONE=YEND:YTWO=YBEG
2350 FOR Y=YMIN TO YMAX+.001 STEP DY
2360 IF (Y YBEG+DY) OR (Y YEND-DY) THEN GOTO 2390
2370 IF Y YONE THEN YONE=Y
2380 IF Y YTWO THEN YTWO=Y
2390 NEXT Y
2400 IF YONE  =YTWO THEN DY=.2*DY:GOTO 2340
2410 IF USEVIEW%=2 THEN COL%=21 ELSE COL%=1
2420 LOCATE 1,COL%
2430 PRINT "Tone " TONE
2440 GET (0,0)-(157,7),MESS3%
2450 CLS
2460 LOCATE 1,COL%
2470 PRINT "Ttwo " TTWO
2480 GET (0,0)-(157,7),MESS4%
2490 CLS:LOCATE 1,COL%
2500 PRINT "Yone " YONE
2510 GET (0,0)-(157,7),MESS5%
2520 CLS:LOCATE 1,COL%
2530 PRINT "Ytwo " YTWO
2540 GET (0,0)-(157,7),MESS6%
2550 IF USEVIEW%=1 THEN VIEW (1,1)-(158,190),,1
2560 IF USEVIEW%=2 THEN VIEW(160,1)-(318,190),,1
2570 IF USEVIEW%=3 THEN VIEW (0,0)-(319,190)
2580 CLS
2590 REM
```

```
2610 WINDOW (TBEG,YBEG)-(TEND,YEND)
2620 IF TONE=TTWO THEN GOTO 2700
2630 YBEGSCREEN=PMAP(YBEG,1)
2640 Y1SCREEN=PMAP(YONE,1)
2650 IF (YBEGSCREEN-Y1SCREEN)<9 THEN YUSE=YBEG+9*(YONE-YBEG)/(YBEGSCREEN-Y1SCREE
N) ELSE YUSE=YONE
2660 PUT (.5*(TBEG+TEND),YUSE),TLABEL,PSET
2670 FOR T=TONE+DT TO (TTWO+.001) STEP DT
2680 LINE (T,YONE)-(T,YONE+.05*(YTWO-YONE)),2
2690 NEXT T
2700 IF YONE=YTWO THEN GOTO 2930
2710 TBEGSCREEN=PMAP(TBEG,0)
2720 T1SCREEN=PMAP(TONE,0)
2730 IF T1SCREEN-TBEGSCREEN < 9 THEN TUSE=TBEG ELSE TUSE=TONE-9*(TONE-TBEG)/(T1S
CREEN-TBEGSCREEN)
2740 PUT (TUSE,.5*(YBEG+YEND)),YLABEL,PSET
2750 LINE (TONE,YONE)-(TTWO,YONE),2
2760 LINE (TONE,YONE)-(TONE,YTWO),2
2770 FOR Y=YONE+DY TO (YTWO+.001) STEP DY
2780 LINE (TONE,Y)-(TONE+.05*(TTWO-TONE),Y),2
2790 NEXT Y
2800 ITRYOLD%=-100
2810 COL%=1:STYLE%=&HFFFF:ITRY%=ITRY0%
2820 GOSUB 2870
2830 GOTO 3000
2840 REM
2850 REM Here starts a small routine to plot a line
2860 REM
2870 IF ITRY%<0 THEN RETURN
2880 IF T1>TBEG THEN TP1=T1 ELSE TP1=TBEG
2890 IF T2<TEND THEN TP2=T2 ELSE TP2=TEND
2900 I%=BEG%(ITRY%):IF TP1 < VALT(I%) THEN TP1=VALT(I%)
2910 I%=BEG%(ITRY%)+NUMENT%(ITRY%)-1:IF TP2 > VALT(I%) THEN TP2=VALT(I%)
2920 TINT=TP1:GOSUB 5400
2930 PSET (TP1,INTERP),COL%
2940 FOR T=TP1 TO TP2 STEP .01*(TP2-TP1)
2950 TINT=T:GOSUB 5400
2960 LINE -(T,INTERP),COL%,,STYLE%
2970 NEXT T
2980 RETURN
2990 REM
3000 COL%=3:STYLE%=&HFFFF:ITRY%=ITRY1%
3010 GOSUB 2870
3020 COL%=2:STYLE%=&H3333:ITRY%=ITRY2%
3030 GOSUB 2870
3040 COL%=2:STYLE%=&HFF00:ITRY%=ITRY3%
3050 GOSUB 2870
3060 REM
3070 REM Produce Messages at bottom of graph
3080 VIEW (0,192)-(319,199)
3090 WINDOW
3100 POSSWAP%=-1
3110 POSSWAP%=POSSWAP%+1
3120 IF POSSWAP%>4 THEN POSSWAP%=0
3130 CLS
3140 IF POSSWAP%=0 THEN PUT (0,0),MESS0%,PSET
3150 IF POSSWAP%=1 THEN PUT (0,0),MESS1%,PSET
3160 IF POSSWAP%=2 THEN PUT (0,0),MESS2%,PSET
3170 IF POSSWAP%=3 THEN PUT (0,0),MESS3%,PSET
3180 IF POSSWAP%=3 THEN PUT (160,0),MESS4%,PSET
3190 IF POSSWAP%=4 THEN PUT (0,0),MESS5%,PSET
3200 IF POSSWAP%=4 THEN PUT (160,0),MESS6%,PSET
3210 FOR I=0 TO 2000:NEXT
3220 REM
3230 REM Get command from keyboard
```

```
3240 K$=INKEY$
3250 IF K$= "" THEN GOTO 3110
3260 IF K$="P" OR K$="p" THEN GOTO 1730
3270 IF K$="E" OR K$="e" THEN GOTO 3340
3280 IF K$="R" OR K$="r" THEN GOTO 1290
3290 IF K$="Z" OR K$="z" THEN GOTO 3370
3300 IF K$="F" OR K$="f" THEN GOTO 3600
3310 GOTO 3240
3320 REM
3330 REM End up Session; set screen nicely
3340 CLS:SCREEN 0,0:WIDTH 80:END
3350 REM
3360 REM Set up zoom
3370 IF NOT FULL% THEN VIEW (160,1)-(318,190),,1 ELSE VIEW
3380 IF FULL% THEN USEVIEW%=3 ELSE USEVIEW%=2
3390 IF FULL% THEN COL%=1 ELSE COL%=21
3400 CLS
3410 LOCATE 2,COL%
3420 INPUT "Zoom factor";Z
3430 IF Z   .0001 THEN GOTO 3410
3440 LOCATE 3,COL%
3450 PRINT "Around t,y values"
3460 LOCATE 4,COL%:PRINT "If Y." YBEGO
3470 LOCATE 5,COL%:PRINT "then use y=f(t)"
3480 LOCATE 6,COL%:INPUT ;TPIVOT,YPIVOT
3490 IF YPIVOT   YBEGO THEN GOTO 3520
3500 ITRY%=0:TINT=TPIVOT:GOSUB 5400
3510 YPIVOT=INTERP
3520 TBEG=TPIVOT-(TPIVOT-TBEGO)/Z
3530 TEND=TPIVOT+(TENDO-TPIVOT)/Z
3540 YBEG=YPIVOT-(YPIVOT-YBEGO)/Z
3550 YEND=YPIVOT+(YENDO-YPIVOT)/Z
3560 CLS
3570 GOTO 2200
3580 REM
3590 REM Change between Full and Half screen
3600 FULL%=NOT FULL%
3610 VIEW:CLS
3620 IF FULL% THEN USEVIEW%=3 ELSE USEVIEW%=1
3630 IF NOT FULL% THEN GOTO 2120
3640 GOTO 2200
3650 REM
3660 REM ---------------------------------------------------------------
3670 REM BEGIN physics section
3680 REM
3690 REM This subroutine interfaces to ODE solvers that step one unit in t
3700 REM INPUT is ido%=0,1,2,3 a pointer to solution technique
3710 REM               delta the step size for t
3720 REM               tphy1 is the initial value of t
3730 REM               yphy1 the initial value for y
3740 REM               yderv1 is the initial value for dy/dt
3750 REM               tphy2/yphy2 respectively is final value of t/y depending
3760 REM               whether tyopt$ is "t" or "y" respectively
3770 REM
3780 REM OUTPUT is nstep% the number of t values generated (1+number of steps)
3790 REM                tempt(1..nstep%) holds t values
3800 REM                tempy(1..nstep%) holds y values
3810 REM
3820 TAC=TPHY1:YAC=YPHY1:DAC=YDERV1:NSTEP%=1
3830 NEQU%=2:YEQU(0)=YAC:YEQU(1)=DAC
3840 IF NODRAG%=1 THEN GOTO 3850
3850 REM
3860 TEMPT(NSTEP%)=TAC
3870 TEMPY(NSTEP%)=YAC
3880 TEMPV(NSTEP%)=DAC
3890 IF TYOPT$="y" THEN GOTO 4370
```

```
3900 REM
3910 REM Section for tyopt$="t" or for tyopt$="y" and y < yphy2
3920 IF TAC >= TPHY2 THEN RETURN
3930 IF NSTEP% >= 400 THEN PRINT "Too many integration steps": RETURN
3940 IF TAC+DELTA > TPHY2 THEN STEPEQU=TPHY2-TAC ELSE STEPEQU=DELTA
3950 GOSUB 4010
3960 GOTO 4330
3970 REM
3980 REM A small subroutine to calculate yac at tac+stepequ
3990 REM TAC is replaced by its new value TAC+STEPEQU
4000 REM by either analytic method or by step of differential equation
4010 IF IDO% <> 0 THEN GOTO 4260
4020 REM
4030 REM place analytic form here
4040 REM if not available place ido%=3 and continue
4050 IF NODRAG%=0 GOTO 4120
4060 YAC=YPHY1+YDERV1*(TAC+STEPEQU)+.5*AODRAG*((TAC+STEPEQU)^2)
4070 DAC=YDERV1+AODRAG*(TAC+STEPEQU)
4080 GOTO 4290
4090 REM
4100 REM Exact solution with drag only valid if initial derivative
4110 REM positive or zero
4120 IF YDERV1 < 0 THEN IDO%=3:GOTO 4260
4130 TAUSCALE=SQR(AODRAG*A2DRAG)
4140 VZERO=YDERV1*A2DRAG/TAUSCALE
4150 CZERO=.5*LOG((1'+VZERO)/ABS(1'-VZERO))
4160 FC=FNCOSH(TAUSCALE*(TAC+STEPEQU)+CZERO)
4170 FS=FNSINH(TAUSCALE*(TAC+STEPEQU)+CZERO)
4180 IF VZERO > 1 THEN GOTO 4220
4190 YAC=YPHY1+LOG(FC/FNCOSH(CZERO))/A2DRAG
4200 DAC=TAUSCALE*FS/(FC*A2DRAG)
4210 GOTO 4290
4220 YAC=YPHY1+LOG(FS/FNSINH(CZERO))/A2DRAG
4230 DAC=TAUSCALE*FC/(FS*A2DRAG)
4240 GOTO 4290
4250 REM
4260 TEQU=TAC:GOSUB 4770
4270 YAC=YEQU(0)
4280 DAC=YEQU(1)
4290 TAC=TAC+STEPEQU
4300 RETURN
4310 REM
4320 REM Next step
4330 NSTEP%=NSTEP%+1
4340 GOTO 3920
4350 REM
4360 REM Section for tyopt$="y"
4370 IF YAC>YPHY2 THEN RETURN
4380 IF NSTEP% >= 400 THEN PRINT "Too many integration steps": RETURN
4390 IF YAC < YPHY2 THEN STEPEQU=DELTA:GOTO 3950
4400 REM
4410 REM Desired y value bracketed, search for correct value
4420 TOLSECT=.01*DELTA
4430 MAXSECT%=30
4440 FSECT1=TEMPY(NSTEP%-1)-YPHY2
4450 FSECT2=TEMPY(NSTEP%)-YPHY2
4460 TSECT1=TEMPT(NSTEP%-1)
4470 TSECT2=TEMPT(NSTEP%)
4480 GOSUB 5870
4490 TEMPT(NSTEP%)=TSECT
4500 TEMPY(NSTEP%)=YAC
4510 TEMPV(NSTEP%)=DAC
4520 RETURN
4530 REM
4540 REM Subroutine called by routine used to solve y(t)=yphy2
```

```
4560 TEQU=TAC
4570 GOSUB 4010
4580 FSECT=YAC-YPHY2
4590 RETURN
4600 REM
4610 REM
4620 REM
4630 REM User supplied routine that is called by ODE solvers
4640 REM For iequ%=0 to nequ%-1 set derivatives of y's wrt t
4650 REM in dyequ(0..nequ%-1)
4660 REM nequ% y variables in yequ(0..nequ%-1)
4670 REM
4680 REM in this case nequ%=2 and
4690 REM for iequ%=0 yequ is y and
4700 REM for iequ%=1 yequ is dy/dt
4710 DYEQU(0)=YEQU(1)
4720 IF NODRAG%=1 THEN GOTO 4750
4730 DYEQU(1)=A0DRAG-A2DRAG*YEQU(1)*ABS(YEQU(1))
4740 RETURN
4750 DYEQU(1)=A0DRAG
4760 RETURN
4770 REM
4780 REM ----------------------------------------------------------------
4790 REM BEGIN numerical analysis section
4800 REM
4810 REM Make 1 step in numerical solution of an ODE using the
4820 REM method selected in ido%
4830 REM ido%=1 Euler
4840 REM ido%=2 2nd order Runge Kutta called Modified Euler method
4850 REM on page 203 of Burden Faires and Reynolds
4860 REM ido%=3 4th order Runge Kutta (p205 Burden,Faires,Reynolds)
4870 REM Also input should be t value in tequ and t step in stepequ
4880 REM and y values in yequ(0..nequ%-1) where input nequ% holds number
4890 REM of y values i.e. number of differential equations
4900 REM On output yequ holds estimate of yequ at tequ+stepequ
4910 REM no other variables are changed
4920 REM user should allocate space for yequ and dyequ earlier
4930 REM
4940 REM Euler's rather simple method
4950 IF IDO%    1 THEN GOTO 5030
4960 GOSUB 4610
4970 FOR IEQU%=0 TO NEQU%-1
4980 YEQU(IEQU%)=YEQU(IEQU%)+STEPEQU*DYEQU(IEQU%)
4990 NEXT IEQU%
5000 RETURN
5010 REM
5020 REM 2nd or 4th order Runge Kutta
5030 DIM K1EQU(20),K2EQU(20),K3EQU(20),YSAVEQU(20)
5040 TSAVEQU=TEQU
5050 GOSUB 4610
5060 IF IDO%=2 THEN FUDEQU=1! ELSE FUDEQU=.5
5070 FOR IEQU%=0 TO NEQU%-1
5080 YSAVEQU(IEQU%)=YEQU(IEQU%)
5090 K1EQU(IEQU%)=STEPEQU*DYEQU(IEQU%)
5100 YEQU(IEQU%)=YEQU(IEQU%)+FUDEQU*K1EQU(IEQU%)
5110 NEXT IEQU%
5120 TEQU=TSAVEQU+FUDEQU*STEPEQU
5130 GOSUB 4610
5140 REM
5150 REM 2nd order Runge Kutta
5160 IF IDO%    2 THEN GOTO 5230
5170 FOR IEQU%=0 TO NEQU%-1
5180 YEQU(IEQU%)=YSAVEQU(IEQU%)+.5*(STEPEQU*DYEQU(IEQU%)+K1EQU(IEQU%))
5190 NEXT IEQU%
5200 GOTO 5270
5210 REM
```

```
5220 REM [illegible]
5230 FOR IEQU%=0 TO NEQU%-1
5240 K2EQU(IEQU%)=STEPEQU*DYEQU(IEQU%)
5250 YEQU(IEQU%)=YSAVEQU(IEQU%)+.5*K2EQU(IEQU%)
5260 NEXT IEQU%
5270 GOSUB 4610
5280 FOR IEQU%=0 TO NEQU%-1
5290 K3EQU(IEQU%)=STEPEQU*DYEQU(IEQU%)
5300 YEQU(IEQU%)=YSAVEQU(IEQU%)+K3EQU(IEQU%)
5310 NEXT IEQU%
5320 TEQU=TSAVEQU+STEPEQU
5330 GOSUB 4610
5340 FOR IEQU%=0 TO NEQU%-1
5350 YEQU(IEQU%)=YSAVEQU(IEQU%)+.1666667*(K1EQU(IEQU%)+2!*(K2EQU(IEQU%)+K3EQU(IE
QU%))+STEPEQU*DYEQU(IEQU%))
5360 NEXT IEQU%
5370 TEQU=TSAVEQU
5380 ERASE K1EQU,K2EQU,K3EQU,YSAVEQU
5390 RETURN
5400 REM
5410 REM
5420 REM Interpolate for various y values given
5430 REM INPUT tint as t value to be interpolated at
5440 REM         itry% to select entry (=0 to ntry%)
5450 REM         itryold% must be set by user to nonsense value
5460 REM         before first call to this routine
5470 REM OUTPUT is interpolated y value in interp
5480 REM
5490 IF ITRY%= ITRYOLD% THEN GOTO 5530
5500 ITRYOLD%=ITRY%
5510 BEGINT%=1
5520 GOTO 5540
5530 IF TINT   TINTOLD THEN GOTO 5510
5540 BIGINT%=BEG%(ITRY%)+BEGINT%-1
5550 FOR RUNINT%=BEGINT% TO NUMENT%(ITRY%)
5560 IF TINT . VALT(BIGINT%) THEN GOTO 5610
5570 BIGINT%=BIGINT%+1
5580 NEXT RUNINT%
5590 I1INT%=NUMENT%(ITRY%)-1
5600 GOTO 5630
5610 I1INT%=RUNINT%-1
5620 IF I1INT%=0 THEN I1INT%=1
5630 BEGINT%=I1INT%+1
5640 TINTOLD=TINT
5650 I1INT%=I1INT%+BEG%(ITRY%)-1
5660 I2INT%=I1INT%+1
5670 FUDINT=(TINT-VALT(I1INT%))/(VALT(I2INT%)-VALT(I1INT%))
5680 INTERP=VALY(I1INT%)*(1-FUDINT)+VALY(I2INT%)*FUDINT
5690 RETURN
5700 REM
5710 REM
5720 REM
5730 REM Bisection technique for solving f(t)=0 given on page 22(algorithm 2.1)
5740 REM of Burden Faires and Reynolds
5750 REM User must supply tolsect - the accuracy required for solution which is
5760 REM returned in tsect with corresponding function value in fsect.
5770 REM User must also supply maxsect% - the maximum number of iterations.
5780 REM (Note accuracy halves each iteration and so one can easily guess
5790 REM how many iterations are necessary)
5800 REM Problem defined by following variables
5810 REM fsect1 : The value of f(t) at t=tsect1
5820 REM fsect2 : the value of f(t) at t=tsect2
5830 REM The User must ensure that fsect1 and fsect2 have opposite sign
5840 REM
5850 REM Finally user must supply a function to calculate f(t)
5860 REM This routine must return f(t) in fsect give t in tsect.
```

```
5870 CTSECT%=0
5880 TSECT=.5*(TSECT1+TSECT2)
5890 GOSUB 4550
5900 IF FSECT=0 THEN RETURN
5910 IF ABS(TSECT2-TSECT1) < 2!*TOLSECT THEN RETURN
5920 CTSECT%=CTSECT%+1
5930 IF CTSECT% >= MAXSECT% THEN PRINT "Too many iterations in bisection algorit
hm":GOTO 3340
5940 IF FSECT1*FSECT > 0 THEN TSECT1=TSECT ELSE TSECT2=TSECT
5950 GOTO 5880
```

NUMDERV1.BAS

```
10 REM This illustrates Numerical Differentiation
20 REM change fny and fndivy to get other examples
30 REM
40 REM ------------------------------------------------------------------------
50 REM BEGIN setup section
60 CLEAR
70 REM
80 REM Set up functions to define functions and derivatives
90 REM Central Derivatives Exact for a Quadratic'
100 DEF FNY(T)=4.9*(T^2)
110 DEF FNDIVY(T)=9.8*T
120 REM Couldnt get fntangy to work with call to fny in it directly so
130 REM SET VALFN=FNY(TZERO)
140 DEF FNTANGY(T,TZERO,DERV)=VALFN+(T-TZERO)*DERV
150 DEF FNFORDIV(T,D)=(FNY(T+D)-FNY(T))/D
160 DEF FNCENTDIVY(T,D)=(FNY(T+D*.5)-FNY(T-D*.5))/D
170 REM
180 REM Arrays for Plot Labels
190 DIM YLABEL(10),TLABEL(10)
200 DIM MESS0%(322),MESS1%(322),MESS2%(322)
210 DIM MESS3%(162),MESS4%(162),MESS5%(162),MESS6%(162)
220 REM
230 REM ------------------------------------------------------------------------
240 REM BEGIN user section
250 REM
260 SCREEN 0,0:WIDTH 80:CLS
270 KEY OFF
280 LOCATE 25,1
290 PRINT "D halves delta P plots R restarts E ends"
300 LOCATE 1,1
310 REM
320 REM Define problem with data from keyboard
330 INPUT "Tzero Delta";TZERO,DELTA
340 REM
350 REM Set and print values of derivatives
360 DERVFOR=FNFORDIV(TZERO,DELTA)
370 DERVCENT=FNCENTDIVY(TZERO,DELTA)
380 DERVREAL=FNDIVY(TZERO)
390 PRINT "tzero" TZERO "delta" DELTA;
400 PRINT "Real    " DERVREAL;
410 PRINT "Forward" DERVFOR;
420 PRINT "Central" DERVCENT
430 REM
440 REM READ COMMAND FROM KEYBOARD
450 K$=INKEY$
460 IF K$="" THEN GOTO 450
470 IF K$="D" OR K$="d" THEN GOTO 540
480 IF K$="P" OR K$="p" THEN GOTO 600
490 IF K$="R" OR K$="r" THEN GOTO 260
500 IF K$="E" OR K$="e" THEN GOTO 2120
510 GOTO 450
520 REM
530 REM Halve interval delta
540 DELTA=.5*DELTA:GOTO 360
550 GOTO 360
560 REM
570 REM ------------------------------------------------------------------------
580 REM BEGIN section to produce screen plots
590 REM
600 SCREEN 0,0:WIDTH 40:CLS
610 INPUT "Plot T range";T1,T2
620 INPUT "Plot Y range";Y1,Y2
630 REM
```

```
640 REM Find range of plots
650 TMIN=T1
660 IF T1 T2 THEN  TMIN=T2
670 TMAX=T2
680 IF T2'T1  THEN TMAX=T1
690 YMIN=Y1
700 FULL%=0
710 IF Y1 Y2 THEN YMIN=Y2
720 YMAX=Y2
730 IF Y2 Y1  THEN YMAX=Y1
740 REM
750 REM Set text strings holding labels for later use
760 CLS
770 SCREEN 1,0
780 LOCATE 1,1
790 PRINT "y"
800 GET (0,0)-(7,7),YLABEL
810 LOCATE 1,1
820 PRINT "t"
830 GET (0,0)-(7,7),TLABEL
840 LOCATE 1,1
850 PRINT "Solid-real Pink dot-forward dash-central"
860 GET (0,0)-(319,7),MESS0%
870 CLS:LOCATE 1,1
880 PRINT "F Flips Full - Half Size"
890 GET (0,0)-(319,7),MESS1%
900 CLS:LOCATE 1,1
910 PRINT "E ends R restarts Z zooms P plots"
920 GET (0,0)-(319,7),MESS2%
930 CLS
940 DT=.101*ABS(T2-T1)
950 DY=.101*ABS(Y2-Y1)
960 TBEG=TMIN-DT:TEND=TMAX+DT:YBEG=YMIN-DY:YEND=YMAX+DY
970 TBEGO=TBEG:TENDO=TEND:YBEGO=YBEG:YENDO=YEND
980 USEVIEW%=1
990 REM
1000 REM Start Plot by deciding on graph axis limits
1010 IF USEVIEW%=1 THEN VIEW (0,0)-(158,190)
1020 IF USEVIEW%=2 THEN VIEW(160,0)-(318,190)
1030 IF USEVIEW%=3 THEN VIEW (0,0)-(319,190)
1040 WINDOW
1050 CLS
1060 DT=.1*ABS(T2-T1)
1070 DY=.1*ABS(Y2-Y1)
1080 TONE=TEND:TTWO=TBEG
1090 FOR T=TMIN TO TMAX+.001 STEP DT
1100 IF (T TBEG+DT) OR (T TEND-DT) THEN GOTO 1130
1110 IF T TONE THEN TONE=T
1120 IF T TTWO THEN TTWO=T
1130 NEXT T
1140 IF TONE  = TTWO THEN DT=.2*DT:GOTO 1080
1150 YONE=YEND:YTWO=YBEG
1160 FOR Y=YMIN TO YMAX+.001 STEP DY
1170 IF (Y YBEG+DY) OR (Y YEND-DY) THEN GOTO 1200
1180 IF Y YONE THEN YONE=Y
1190 IF Y YTWO THEN YTWO=Y
1200 NEXT Y
1210 IF YONE  =YTWO THEN DY=.2*DY:GOTO 1150
1220 IF USEVIEW%=2 THEN COL%=21 ELSE COL%=1
1230 LOCATE 1,COL%
1240 PRINT "Tone " TONE
1250 GET (0,0)-(157,7),MESS3%
1260 CLS
1270 LOCATE 1,COL%
1280 PRINT "Ttwo " TTWO
1290 GET (0,0)-(157,7),MESS4%
```

```
1300 CLS:LOCATE 1,COL%
1310 PRINT "Yone " YONE
1320 GET (0,0)-(157,7),MESS5%
1330 CLS:LOCATE 1,COL%
1340 PRINT "Ytwo " YTWO
1350 GET (0,0)-(157,7),MESS6%
1360 IF USEVIEW%=1 THEN VIEW (1,1)-(158,190),,1
1370 IF USEVIEW%=2 THEN VIEW(160,1)-(318,190),,1
1380 IF USEVIEW%=3 THEN VIEW (0,0)-(319,190)
1390 CLS
1400 REM
1410 REM PLOT CURVES: SET UP WINDOW TO SCALE CORRECTLY
1420 WINDOW (TBEG,YBEG)-(TEND,YEND)
1430 IF TONE=TTWO THEN GOTO 1510
1440 YBEGSCREEN=PMAP(YBEG,1)
1450 Y1SCREEN=PMAP(YONE,1)
1460 IF (YBEGSCREEN-Y1SCREEN) 9 THEN YUSE=YBEG+9*(YONE-YBEG)/(YBEGSCREEN-Y1SCREE
N) ELSE YUSE=YONE
1470 PUT (.5*(TBEG+TEND),YUSE),TLABEL,PSET
1480 FOR T=TONE+DT TO (TTWO+.001) STEP DT
1490 LINE (T,YONE)-(T,YONE+.05*(YTWO-YONE)),2
1500 NEXT T
1510 IF YONE=YTWO THEN GOTO 1710
1520 TBEGSCREEN=PMAP(TBEG,0)
1530 T1SCREEN=PMAP(TONE,0)
1540 IF T1SCREEN-TBEGSCREEN   9 THEN TUSE=TBEG ELSE TUSE=TONE-9*(TONE-TBEG)/(T1S
CREEN-TBEGSCREEN)
1550 PUT (TUSE,.5*(YBEG+YEND)),YLABEL,PSET
1560 LINE (TONE,YONE)-(TTWO,YONE),2
1570 LINE (TONE,YONE)-(TONE,YTWO),2
1580 FOR Y=YONE+DY TO (YTWO+.001) STEP DY
1590 LINE (TONE,Y)-(TONE+.05*(TTWO-TONE),Y),2
1600 NEXT Y
1610 COL%=1:STYLE%=&HFFFF:ITRY%=0
1620 GOSUB 1670
1630 GOTO 1780
1640 REM
1650 REM Here starts a small routine to plot a line
1660 REM
1670 IF ITRY% 0 THEN RETURN
1680 IF T1 TBEG THEN TP1=T1 ELSE TP1=TBEG
1690 IF T2 TEND THEN TP2=T2 ELSE TP2=TEND
1700 TINT=TP1:GOSUB 2470
1710 PSET (TP1,INTERP),COL%
1720 FOR T=TP1 TO TP2 STEP .01*(TP2-TP1)
1730 TINT=T:GOSUB 2470
1740 LINE -(T,INTERP),COL%,,STYLE%
1750 NEXT T
1760 RETURN
1770 REM
1780 COL%=3:STYLE%=&HFFFF:ITRY%=1
1790 GOSUB 1670
1800 COL%=2:STYLE%=&H3333:ITRY%=2
1810 GOSUB 1670
1820 COL%=2:STYLE%=&HFF00:ITRY%=3
1830 GOSUB 1670
1840 REM
1850 REM Produce Messages at bottom of graph
1860 VIEW (0,192)-(319,199)
1870 WINDOW
1880 POSSWAP%=-1
1890 POSSWAP%=POSSWAP%+1
1900 IF POSSWAP% 4 THEN POSSWAP%=0
1910 CLS
1920 IF POSSWAP%=0 THEN PUT (0,0),MESS0%,PSET
1930 IF POSSWAP%=1 THEN PUT (0,0),MESS1%,PSET
1940 IF POSSWAP%
```

```
[illegible] IF POSSWAP%=2 THEN PUT (0,0),MESS2%,PSET
1950 IF POSSWAP%=3 THEN PUT (0,0),MESS3%,PSET
1960 IF POSSWAP%=3 THEN PUT (160,0),MESS4%,PSET
1970 IF POSSWAP%=4 THEN PUT (0,0),MESS5%,PSET
1980 IF POSSWAP%=4 THEN PUT (160,0),MESS6%,PSET
1990 FOR I=0 TO 2000:NEXT
2000 REM
2010 REM Get command from keyboard
2020 K$=INKEY$
2030 IF K$= "" THEN GOTO 1890
2040 IF K$="P" OR K$="p" THEN GOTO 600
2050 IF K$="E" OR K$="e" THEN GOTO 2120
2060 IF K$="R" OR K$="r" THEN GOTO 260
2070 IF K$="Z" OR K$="z" THEN GOTO 2150
2080 IF K$="F" OR K$="f" THEN GOTO 2380
2090 GOTO 2020
2100 REM
2110 REM End up Session; set screen nicely
2120 CLS:SCREEN 0,0:WIDTH 80:END
2130 REM
2140 REM Set up zoom
2150 IF NOT FULL% THEN VIEW (160,1)-(318,190),,1 ELSE VIEW
2160 IF FULL% THEN USEVIEW%=3 ELSE USEVIEW%=2
2170 IF FULL% THEN COL%=1 ELSE COL%=21
2180 CLS
2190 LOCATE 2,COL%
2200 INPUT "Zoom factor";Z
2210 IF Z   .0001 THEN GOTO 2190
2220 LOCATE 3,COL%
2230 PRINT "Around t,y values"
2240 LOCATE 4,COL%:PRINT "If Y ." YBEGO
2250 LOCATE 5,COL%:PRINT "then use y=f(t)"
2260 LOCATE 6,COL%:INPUT ;TPIVOT,YPIVOT
2270 IF YPIVOT   YBEGO THEN GOTO 2300
2280 ITRY%=0:TINT=TPIVOT:GOSUB 2470
2290 YPIVOT=INTERP
2300 TBEG=TPIVOT-(TPIVOT-TBEGO)/Z
2310 TEND=TPIVOT+(TENDO-TPIVOT)/Z
2320 YBEG=YPIVOT-(YPIVOT-YBEGO)/Z
2330 YEND=YPIVOT+(YENDO-YPIVOT)/Z
2340 CLS
2350 GOTO 1010
2360 REM
2370 REM Change between Full and Half screen
2380 FULL%=NOT FULL%
2390 VIEW:CLS
2400 IF FULL% THEN USEVIEW%=3 ELSE USEVIEW%=1
2410 IF NOT FULL% THEN GOTO 930
2420 GOTO 1010
2430 REM
2440 REM ------------------------------------------------------------------
2450 REM BEGIN section to calculate functions for display
2460 REM
2470 IF ITRY%=0 THEN INTERP=FNY(TINT):RETURN
2480 VALFN=FNY(TZERO)
2490 IF ITRY%=1 THEN INTERP=FNTANGY(TINT,TZERO,DERVREAL):RETURN
2500 IF ITRY%=2 THEN INTERP=FNTANGY(TINT,TZERO,DERVFOR):RETURN
2510 IF ITRY%=3 THEN INTERP=FNTANGY(TINT,TZERO,DERVCENT):RETURN
2520 INTERP=0:RETURN
```