UCSB Math TI-85 Tutorials: Basics

If your calculator screen doesn't show anything, try adjusting the contrast according to the instructions on page 3, or page I-3, of the calculator manual. You should read the "Getting Started" chapter of your calculator manual through page 9, and the pages I-1 to I-14 Chapter 1, "Operating the TI-85." But if you haven't mastered those sections, don't worry about it; read this page, and then go back to the manual. Most of the stuff there and here becomes completely clear only after you've read it more than once and tried it out on your calculator.

These tutorial sheets are only about topics of direct use in UCSB Math courses. There's a lot more to the TI-85 than is dealt with here. This sheet concerns some basic points on using your calculator. Other sheets are about running and editing calculator programs. Your TA may provide some programs for you in discussion—you will download them into your calculator, so bring the wire that came with it. Your TA will also go over the use of the program editor.

Names: Variables, programs, lists, and sundry other things on the TI-85 have names, which are strings of letters and numbers with the first being a letter. To type a letter, first press the blue ALPHA key, then the letter key (the letters are in blue above the keys). To type a lower-case letter, press 2nd, then ALPHA, then the letter. Two ALPHA's in succession put the calculator in *alpha lock mode*, useful for typing a sequence of letters. When in alpha lock mode, 2nd ALPHA and ALPHA switch back and forth between upper and lower case. ALPHA when in upper-case alpha lock mode takes it out of alpha mode. You can tell when the calculator is in alpha mode because the cursor becomes a white-on-black A. The x-VAR key is a shortcut for a lower-case x. Also included with the "letters" are the equals sign = and the blank space (symbolized on the keypad by ␣).

Equations: Let's enter a simple equation into the calculator variable F. Type this

```
F=x^3-1
```

The actual sequence of keypresses to do this is

```
ALPHA  F  ALPHA  =  x-VAR  ^  3  -  1
```

Now press ENTER. You should see the word "Done" on the screen. Next, type

```
3→x
```

This time, the actual key sequence is

```
3   STO▸   x-VAR
```

Pressing ENTER stores the number 3 into variable x. Type F and press ENTER. The contents of F are a formula that involves x; the value stored in x is used to evaluate the formula and the result of the evaluation is displayed. You should see 26, i.e., $3^3 - 1$. Now store 4 into x, and evaluate F again—you'll get 63, which is $4^3 - 1$.

When you type F and press ENTER, you get the result of *evaluating* whatever is stored in F. But if you want to see *literally* what is stored in F, you can use the RCL key (secondary function of the STO▸ key). Type 2nd RCL, then F, then ENTER.

The colon : The colon : acts as a command separator. With it, you can put two or more commands on the same line. In the example above, we could have typed the single line

```
F=x^3-1:3→x:F
```

pressed ENTER, and gotten the same result.

Useful keys: 2nd Ans always holds the last value displayed on the screen. If you omit an initial operand, e.g., type + 3, the calculator will insert Ans. 2nd ENTRY always holds the last line *YOU* typed. It's useful for correcting mistakes: you can get back the last line, then edit it using the arrow keys, DEL, and the insert key, 2nd INS (secondary function of DEL).

Menus: Many keys produce *menus* of further operations, which are displayed at the bottom of the screen. For example, 2nd MATH brings up a menu. One of the menu items is MISC; select it by pressing the button immediately below (the one labelled F5). The result is a *secondary menu*. A small arrowhead ▸ at the right indicates that still more menu items are available; press MORE to bring these into view. Let's use $\sqrt[x]{}$, the one at the F4 key: it simply puts that symbol on the calculator screen. Type this line

```
3 ˣ√ 2
```

When you press ENTER, the calculator will display $\sqrt[3]{2}$.

One press of EXIT suppresses the secondary menu, and another removes the menu altogether. When a secondary menu is visible, you can still select an item from the primary menu by pressing 2nd before the menu key.

Brackets [and] signal menu choices in these tutorials. We just did 2nd MATH [MISC].

Some special characters are accessed through menus; e.g, use 2nd LIST to get braces { and }, 2nd TEST to get ≤.

Reserved names: The TI-85 has some *reserved names*, used for its internal functions or for some built-in constants. Page 8-2 of the calculator manual lists the built-in constants. You can refer to these names, but you can't change them. Unfortunately, some of the names, such as h, c, g, k, and u, are common letters that you're liable to try to use for other purposes. You can't, and you'll get an error message if you try. Other kinds of reserved names, e.g., calculator function names like sin, are listed in Appendix A. On page A-22 are also listed "variables used by the TI-85." These are things that you can change, but many of the calculator functions also change them as side effects. For example, x and y are used by the graphing functions of the calculator.

Modes: The calculator has a variety of modes. Type 2nd MODE and you'll see a screenful. When you position the cursor on one of these and press ENTER, that mode will be set. For example, if you move the cursor to the 4 on the line beginning with Float and press ENTER, the calculator will now display numbers using only 4 digits to the right of the decimal point. (It will still compute with all 12 digits, however.) Press EXIT to get back to the home screen, and try it out. Put the mode back to Float to get all digits available.

doner@math.ucsb.edu

UCSB Math TI-85 Tutorials: Derivatives

Derivatives: The TI-85 can generate estimates of the derivative of a function at a point; i.e., the slope of its graph there. There are several possibilities for the definition of the derivative; we'll use

$$f'(x) = \lim_{H \to 0} \frac{f(x+H) - f(x-H)}{2H}$$

With small values of H, the quotient in this formula should be a good approximation to $f'(x)$.

Notice we don't calculate $f(x)$ at all, but rather $f(x+H)$ and $f(x-H)$. The TI-85 has a feature to help in situations like this: the `evalF` function, which is used to evaluate an expression with a temporary change to a variable occuring in the expression, and has the syntax

`evalF(`*expression, variable, temporary value*`)`

For example, `evalF(A^3,A,5)` yields 125, but doesn't change whatever value may be in `A`. (For convenience, you can get `evalF(` as the first menu choice after `2nd CALC`.)

Let's put the formula for approximating the derivative into a calculator variable:

```
DR=(evalF(F,x,x+H)-evalF(F,x,x-H))/(2H)
```

So if we put an expression for a function f into `F`, the value of x into `x`, and some small number into `H`, just evaluating `DR` will give us the corresponding approximation to $f'(x)$.[1]

We'll illustrate the procedure by finding $f'(.7)$ where $f(x) = \cos(3x)$. First, we put the defining expression for the function f into the calculator variable `F`, and .7 in the calculator variable `x`:

```
F=cos (3x)
.7→x
```

Then we give `H` the value 1 and evaluate `DR`:

```
1→H
DR
```

If you've done everthing correctly, you should see

```
-.121816112779
```

which is

$$\frac{f(.7+1) - f(.7-1)}{2}$$

If you don't, look for a mistake.[2] If you see correct digits, but not all of them, read the section on Modes in the Basics tutorial.

This is far from a good estimate of $f'(.7)$, because `H` is much too large. Type the line

```
H/2→H:DR
```

and press `ENTER`. (The colon `:` acts as a command separator; with it, you can put two or more commands on the same line. Here, the first command divides `H` by 2, and the second evaluates `DR`.) You should see the result of re-evaluating `DR` with a value of `H` just one-half what it was before—in this case, the new value of `H` is 0.5. Press `ENTER` again, and this time you'll get the result obtained with $H = .25$. Continue pressing `ENTER` until you're satisfied you have a sufficiently accurate answer.

This works because when you just press `ENTER` without typing anything else, the calculator simply re-executes the previous line (i.e., it executes the contents of `2nd ENTRY`). Of course, if you type anything else, and then want to continue, you'll have to retype `H/2→H:DR`.

Accuracy: But how do you tell when your answer is "sufficiently accurate"? Decide how many digits you want accurately, and re-evaluate `DR` with successively smaller step-sizes until that number of digits appears to have *stabilized*—the ones of interest aren't changing any more as you reduce the step-size further. Note that for this to work, it's best to reduce the step size *geometrically*, e.g., by dividing it by 2 or 10 each time.

To illustrate, here is a table of values obtained by evaluating `DR` with a geometrically decreasing step size:

H	Result	Change
1	-.121816112779	
.5	-1.72209403124	-1.600…
.25	-2.35358784929	-.63149…
.125	-2.52935902283	-.17577…
.0625	-2.57448114783	-.04512…
.03125	-2.58583636608	-.01135…
.015625	-2.58867985400	-.00284…
.0078125	-2.58939101892	-.00071…
.00390625	-2.58956882847	-.00017…

Digits are underlined once it is *apparent* that they are stable. In the last line, "-2.589" are all stable digits—however, if this result is to be reported to 4 significant digits, you should round it off, noting that the next digit is at least a "5", and say "-2.590". "Stability" isn't quite the same thing as "accuracy to a certain number of digits".

An important advantage arises from looking at a sequence of answers generated from a geometric sequence of parameter values. If you just make a single calculation, you really have little idea how many digits are correct. Of course, the basic functions of the calculator such as `cos` are programmed to produce all 12 digits correctly. But when you're executing an approximate algorithm, such as the one `DR` represents, you don't get an indication of accuracy from a single evaluation with a single parameter value. On the other hand, if you have a whole series of evaluations with a geometrically changing parameter, and you can see how some of the digits in the answer are stabilizing, you may infer that the stable digits are actually correct digits. Stabilization in this manner is not a sure thing, but it is usually a good indication of correctness.

[1] The `DR` equation is long and hard to type correctly. Watch those parentheses! You can look at `DR` to check it by doing `2nd RCL DR`. (RCL is the secondary function of the `STO▸` key.) To edit it, type `DR= 2nd RCL DR`, then make your changes and finally press `ENTER`. Once you get it right, you'll never have to change `DR` again, because estimating other derivatives only requires changes to `F`, `x`, and `H`.

[2] If you get $-.001917783535$, it's because your calculator is in Degree mode instead of Radian mode.

doner@math.ucsb.edu

UCSB Math TI-85 Tutorials: Differential Equations

Differential Equations and Euler's method: Here's a very simple initial value problem, consisting of a differential equation for an unknown function y and a value for that function at a single point:

$$y' = -.2y, \qquad y(0) = 1$$

This problem can be solved numerically using Euler's method. Euler's method is based on the approximation

$$y(t + \Delta t) = y(t) + \Delta t\, y'(t)$$

(Important: we customarily use t for the independent variable in our differential equations problems instead of x.) The numerical solution via Euler's method is approximate, and usually gets more accurate as the step size Δt is reduced. In numerical work, the step size is often denoted by h or H—we'll use H, because in the TI-85, `h` is reserved for a special constant.

Let's apply Euler's method to the problem above. Suppose we'd like to know the value of $y(2)$, and we think a step size of .1 will be good enough. On the TI-85, we'll use `Y` instead of `y`, because the calculator sometimes uses `y` internally. What we need to do is store 0 into `t`, 1 into `Y`, .1 into `H`, and then repeat the steps `Y+H*(-.2Y)→Y` and `t+H→t` 20 times. Then `t` will have 2 in it, and `Y` will have our estimate for $y(2)$.

For the sake of generality—isolating those parts of this process that are specific to each problem from those which stay the same—we will put the expression for y' in a calculator equation, instead of typing it directly into the calculation. So do these steps, pressing `ENTER` after each line:

```
F=-.2*Y
.1→H
1→Y
0→t
```

What we want to do next is set up a single line that will do all the rest, so we can repeat it by pressing `ENTER`. Type this:

```
Y+H*F→Y:t+H→t:{t,Y}
```

You can put more than one command into an entry by separating them with a `:`. (The `{t,Y}` at the end is a trick to get the calculator to display two things on the same line—you could also use `[t,Y]`, or `(t,Y)`.) Now press `ENTER` 20 times, until the value in `t` is 2. The corresponding `Y` is the answer. If you did it correctly, you'll see

```
{2 .667607971755}
```

This number is not far from the exact answer, $e^{-.4} = .670320046036$, but it isn't very close either. A better answer would be obtained with a smaller `H` and correspondingly more steps. Try repeating the whole process, but with `.02→H` and pressing `ENTER` 100 times.

Lists: The TI-85 operates with lists of values as well as with single values. A list is entered with `{` and `}` symbols, which are found by first typing `2nd LIST`, and then using the two menu keys to get the brackets. For example

```
{2,4,6}→M
```

stores the indicated list of three numbers into the variable `M`. Then `M^2` displays the list of the squares of these numbers, `sin M` displays the list of their sin's, and so on. (When not all of the list fits on the screen, the TI-85 signals this by showing three dots ... where the missing part is; the arrow keys are used to scroll those parts into view.) You must use commas to separate list elements when you enter them, but the TI-85 uses spaces instead when it displays a list. Try entering `2*{3,4}`, then `{2,3}*{4,5}`, and finally `{2,3}*{4,5,6}`. The last gives an error message.

You can refer to the individual elements of lists: with `M` as above, `M(1)` is 2, `M(2)` is 4, and `M(3)` is 6. Doing

```
3→M(2)
```

changes `M` to `{2,3,6}`.

Systems of equations: What if you have a system of differential equations to deal with? For example,

$$\begin{aligned} y' &= -x, & y(0) &= 1 \\ x' &= y, & x(0) &= 0 \end{aligned}$$

is a system with solution $y(t) = \cos t$, $x(t) = \sin t$. You could use two calculator equations named, say, `F` and `G`, two variables `Y` and `X`, and a longer command to repeat. This would probably be o.k. for two equations, but the method rapidly gets cumbersome as we go to problems with three or four simultaneous equations. Instead, we can put everything in lists. `Y` will be a list of two components, the first corresponding to y, and the second to x. Think of it as adopting this translation from mathematical symbols to calculator variables:

$$\begin{aligned} y &\implies \texttt{Y(1)} \\ x &\implies \texttt{Y(2)} \end{aligned}$$

To solve our example, first do this to initialize everything:

```
F={-Y(2),Y(1)}
.1→H
{1,0}→Y
0→t
```

Then type[3]:

```
Y+H*F→Y:t+H→t:Disp t:Y
```

Press `ENTER` repeatedly. `t` and `Y` are displayed on separate lines, and when `t` is 1, we'll see the solution for $y(1)$, $x(1)$.

Obtaining greater accuracy is mainly a matter of using a smaller value for `H`, and pressing `ENTER` correspondingly more times to reach a given value for `t`.

[3] This line is too long to fit on one physical line across the screen, but don't worry about it, a logical line can be longer than a physical line, and the calculator just wraps around. We use `Disp t` because the trick of using `{t,Y}` to display both `t` and `Y` on the same line won't work if `Y` is a list. The required space in `Disp t` is the `ALPHA`-mode of the `(-)` key, the ␣ symbol, second from the right on the bottom row.

doner@math.ucsb.edu

UCSB Math TI-85 Tutorials: Programs

Running programs: To run a program, simply type its name on the main screen, and press ENTER. When a program is running, a group of pixels in the upper right corner of the screen will flash on and off, appearing to be moving. To abort a program that is taking too long (maybe it's in an endless loop?), press the ON key. (Some programs are designed to run indefinitely, so aborting with the ON key is the only way to stop them.) The ON key will also abort other operations, e.g., graph plotting.

Typing PRGM and choosing [NAMES] produces a menu of available programs. Choosing one puts its name on the main screen. Then you can press ENTER to run it.

Program Editing: You will need to consult the manual for a complete treatment of this subject; this tutorial only covers the major points and some helpful tips.

To edit an existing program or create a new one, press PRGM. There are two menu choices. [NAMES] just produces a menu of existing program names; it's only useful for getting the name of a program onto the screen. [EDIT] puts the calculator into program edit mode, and displays the menu of exiting program names. You can type a name of a program, or select a name from the menu, and then press ENTER. If there already exists a program of that name, its text will be read into to the edit buffer, but if it's an altogether new name, the buffer will be empty and you'll simply see a single line with a colon : —you can then begin to type program code.

Program commands are just like the commands you can type directly on the main screen. After you type a line, press ENTER and the cursor will move to a new line. To move from line to line for editing, or to move the cursor within a line, use the arrow keys. The DEL key deletes the character the cursor is on, and 2nd INS allows you to insert additional characters.

The [I/O] menu choice presents a list of input-output commands; choosing one simply types that command into the program at the current cursor position. Similarly, the [CTL] menu choice gives a list of control commands. [INSc] inserts a blank line ahead of the line the cursor is on. [DELc] deletes the line the cursor is on. [UNDEL] inserts the last line deleted ahead of the line the cursor is on. So if you want to move a line from one place to another, you first delete it with [DELc], then you move the cursor to the desired place and choose [UNDEL].

The functions of the various input-output and control commands are discussed in the calculator manual. They're not unlike those in the BASIC programming language.

A colon : is a command separator; with it, you can have two or more commands on the same line (e.g., so that you can fit all the commands of interest in the tiny window). If you position the cursor at the end of a line and press DEL, you'll delete the newline and thus combine two lines. You can separate the commands by inserting a colon with 2nd INS :. To break a line in two, position the cursor where you want the break, and type 2nd INS ENTER.

It's easy to get one or more blank lines, with just a colon, at the end of your program. They don't usually do any harm, but here's how to delete them: put the cursor at the end of the previous line, and press DEL until all the extra lines are gone.

Making copies: Usually, if you want to change a program your teacher supplied, it's better to edit a copy of the program instead of the original. That way, if you make a mistake, you can always go back to the original. Here's how to make a copy of a program: press the PRGM key, then choose EDIT. Type an altogether new name for the program, and press ENTER. You'll be in Edit mode, with the cursor on a line beginning with a colon. Now type 2nd RCL, and then type the name of the program you want to copy (or press PRGM to see the list of programs and pick one with a menu key). When you press ENTER, the entire text of the program you named will be copied into your new program. Edit this as you please, and when you EXIT, you'll have a new program with the name you chose.

Remember: in program Edit mode, you can use 2nd RCL to retrieve the text of any other program into the one you're editing.

Standard Distribution: The Math Department has a standard distribution of TI-85 programs. These include DESetUp, EULER, MEULER, and SIRSetUp, all of which are described in these tutorial sheets. Your teacher or TA will provide the programs, and show you how to copy programs from one calculator to another. Edit the programs as you please, but edit copies, not the originals!

doner@math.ucsb.edu

UCSB Math TI-85 Tutorials: Programming Euler's method

Please read the Differential Equations tutorial before this one.

Program EULER: This program implements Euler's method for numerically solving differential equations, in particular initial value problems of the form:

$$y' = f(t, y), \qquad y(a) = b$$

Here, f is a given function of t and y, where t is the independent variable and y the dependent.

The design of the EULER program is based on the discussion at the beginning of the Differential Equations tutorial, where an equation was solved by setting up some initializations, and then repeatedly executing

```
Y+H*F→Y:t+H→t:(t,Y)
```

Program EULER contains the While ... End loop control construction, so the program itself controls the number of steps. The commands between While and End will be executed until the condition on the While line becomes false. Here is the text of the EULER program:

```
EULER                    • Prog ram
  :YI→Y
  :tI→t
  :(tF-tI)/N→H
  :While t<tF
  :Y+H*F→Y
  :t+H→t
  :End
  :Y
```

It's necessary to set more things initially, but you aren't in the position of pressing ENTER hundreds or thousands of times; it only takes one press to run the program. The EULER program presumes the following are already set:

- the equation for the derivative of Y in F,
- initial values for Y and t in YI and tI,
- a final value for t in tF,
- a desired number of steps in N.

The program calculates a step size H so that exactly N steps will cover the distance from tI to tF. Starting from t = tI, it then proceeds to do as many Euler's method steps as necessary to make t ≥ tF (that's N steps, of course), and finally displays Y.

Example: The *logistic equation* arises in modeling of population growth. Here is a particular problem involving a logistic equation:

$$y' = .1y\left(1 - \frac{y}{1000}\right), \quad y(0) = 100; \qquad \text{Find } y(25)$$

It's often more convenient to put initialization lines in a program of their own. This makes it convenient to re-run them, with changes if necessary. The following program sets things up for EULER to solve the logistic equation problem:

```
DESetUp                  • Prog ram
  :F=.1Y(1-Y/1000)
  :0→tI
  :25→tF
  :100→YI
  :50→N
```

Run DESetUp, then run EULER. If you did it correctly, you should see

```
568.968766939
```

For other differential equations, just edit a copy of DESetUp appropriately—you should only need to change the definition of F and some of the numbers. [4]

If you want more accuracy, just type

```
N*2→N:EULER
```

and press ENTER one or more times. Each time, EULER is run with N doubled, and therefore the step size cut in half. Each time gives better accuracy, but takes twice as long.

Program EULER can also be used to solve systems of differential equations, in the manner explained in the last section of the Differential Equations tutorial.

Example: To solve the same system as in the Differential Equations tutorial,

$$\begin{aligned} y' &= -x, & y(0) &= 1 \\ x' &= y, & x(0) &= 0 \end{aligned}$$

with the same translation to calculator variables:

$$\begin{aligned} y &\Longrightarrow \texttt{Y(1)} \\ x &\Longrightarrow \texttt{Y(2)} \end{aligned}$$

we first do these initializations

```
F={-Y(2),Y(1)}
{1,0}→YI
1→tF
5→N
```

then type

```
N*2→N:EULER
```

and press ENTER repeatedly. You'll see values for Y obtained with N = 10, 20, 40, ... If you did everything correctly, the answers will converge toward cos 1 and sin 1.

You may find it convenient to create a set-up program, similar to DESetUp, to handle the initializations.

[4] To make a copy: press PRGM [EDIT], type an unused name for the copy—AB, perhaps—press ENTER, and you'll see a line with just a colon. Do 2nd RCL, and then type DESetUp, or press PRGM and choose [DESetUp] from the menu. When you press ENTER, program DESetUp is copied to your new program. Make your changes, and press EXIT when done.

doner@math.ucsb.edu

UCSB Math TI-85 Tutorials: Improving on Euler

Program MEULER: Euler's method can be improved upon considerably at very little computational expense. The Modified Euler's Method uses Euler's method to obtain an initial estimate for $y(t+\Delta t)$, uses that to get an initial estimate of $y'(t+\Delta t)$, and uses the average of this and the originally calculated $y'(t)$ to update y. Where Euler's Method for an equation $y' = f(t,y)$ is based on the equation

$$y(t+\Delta t) = y(t) + \Delta t\ f(t,y),$$

the Modified Euler's Method is based on

$$\begin{aligned} y_{\text{guess}} &= y(t) + \Delta t\ f(t,y), \\ y(t+\Delta t) &= y(t) + \Delta t\ \tfrac{1}{2}(f(t,y(t)) + f(t+\Delta t, y_{\text{guess}})) \end{aligned}$$

Program MEULER implements this method.

```
MEULER                  • Pro9ram
  :YI→Y
  :tI→t
  :(tF-tI)/N→H
  :While t<tF
  :Y→Yold
  :F→Yp
  :Y+H*Yp→Y
  :t+H→t
  :(Yp+F)/2→Yp
  :Yold+H*Yp→Y
  :End
  :Y
```

Program MEULER is interchangeable with EULER, but produces accurate answers more quickly. For a given N, it takes twice as long to run, but still saves time when you're after a certain level of accuracy.

For a given value of N, MEULER does about twice as many calculations as EULER. Try this experiment: using a simple system, such as the sin and cos one given in the "Programming Euler's Method" tutorial, run MEULER with some N, and EULER with $2*$N. About the same amount of computational labor, but how do they compare on accuracy? Double N and repeat the experiment.

Example: The SIR model, featured in *Calculus in Context* and many other books, represents the spread of disease through a population. S stands for the number of "susceptibles," I for the "infecteds," and R for the "recovereds." In this case, we're considering a population of 50000, 2100 of whom are infected at time 0, and 2500 already recovered (and therefore immune).

$$\begin{aligned} S' &= -.0001SI \\ I' &= .00001SI - I/14 \\ R' &= I/14 \\ S(0) &= 45400,\ I(0) = 2100,\ R(0) = 2500. \end{aligned}$$

We can use MEULER (or EULER) to solve this system. We'll use the obvious translation of the mathematical symbols to calculator variables.

$$\begin{aligned} S &\Longrightarrow \text{Y(1)} \\ I &\Longrightarrow \text{Y(2)} \\ R &\Longrightarrow \text{Y(3)} \end{aligned}$$

With that, we can use the following initializations to find, say, the solution for $t = 10$:

```
F={-.00001*Y(1)*Y(2),
  .00001*Y(1)*Y(2)-Y(2)/14,Y(2)/14}
{45400,2100,2500}→YI
0→tI
10→tF
10→N
```

Type N*2→N:MEULER (or N*2→N:EULER) and press ENTER. With MEULER, you should see

```
{13349.8822803 2536…
```

The three dots … indicate that part of the answer is off screen; use the arrow keys to scroll it into view. This answer was obtained with ½ day steps, corresponding to $N = 20$. Press ENTER again to repeat the calculation with a smaller step size.

The equation for F was long and rather clumsy to type, and perhaps hard to understand later. We can use a trick to improve readability: establish the symbol translations as calculator equations. Putting it all in a set-up program, we get

```
SIRSetUp              • Pro9ram
  :S=Y(1)
  :I=Y(2)
  :R=Y(3)
:F={-.00001*S*I,.00001*S*I-I/14,I/14}
  :{45400,2100,2500}→YI
  :0→tI
  :10→tF
  :10→N
```

Run SIRSetUp, then type N*2→N:MEULER and press ENTER.

doner@math.ucsb.edu

UCSB Math TI-85 Tutorials:

Graphing DE solutions (sin–cos equations)

Graphing Solutions of Differential Equations: The TI-85 has a built-in module for graphing the (numerical) solutions of differential equations. We could actually write programs that would do this, but the built-in module is faster, more accurate, and more flexible than anything we could easily create. The built-in module uses a numerical method much better than Euler's; unfortunately, its mathematical sophistication precludes discussion of it in this tutorial. A downside is that the built-in module uses a fixed set of variable names, `Q1, Q2,..., Q9`, so you must forego using the names natural to your problem. The big advantage of this module is that you don't have to write programs, and you get both graphical and numeric solutions.

Example: We'll illustrate the use of the module by solving the system

$$\begin{aligned} y' &= -x, & y(0) &= 1 \\ x' &= y, & x(0) &= 0 \end{aligned}$$

The exact solution for this system is

$$\begin{aligned} y &= \cos(t) \\ x &= \sin(t) \end{aligned}$$

The first step is to place the calculator in its *differential equation graphing mode*. From the home screen, use the `MODE` command (the secondary function of the `MORE` key), then use the arrow keys to move the cursor to the line

```
Func   Pol   Param   DifEq
```

Move the cursor over to the `DifEq` item and press `ENTER`. Use `EXIT` to return to the home screen. Press `GRAPH`. As you see, the menus are different from those in ordinary graphing mode. Select the menu item `Q'(t)=`. If you see equations already there, use the `[DELf]` menu item repeatedly to delete them. Now type a new set of equations

```
Q'1=-Q2
Q'2=Q1
```

Pressing `ENTER` at the end of a line, or moving the cursor below the last line, causes the calculator to begin a new equation. You don't have to press `ENTER` after the last equation, but it doesn't hurt if you do. As you have probably inferred, we replaced y by `Q1`, and x by `Q2`.

Next, you must set the initial conditions. Choose the `[INITC]` menu item, and set up

```
QI1=1
QI2=0
```

You also need to set the ranges—choose the `[RANGE]` item, and set these values:

```
tMin=0
tMax=10
tStep=.2
tPlot=0
xMin=0
xMax=10
xScl=1
yMin=-1
yMax=1
yScl=.2
difTol=.001
```

Now you're ready—choose the `[GRAPH]` menu item and watch it go. When it finishes, you can press `CLEAR` to remove the menu bar from the screen, and `GRAPH` or `EXIT` (once) to bring it back. Try the `[TRACE]` item; you can move the cursor forward from the beginning with the right arrow, and you can use the up and down arrows to move it from one graph to another (the number in the upper right of the screen tells you which curve it's on).

`tMin, tMax` govern the range of `t`-values for which the calculations are made;

`tStep` determines the `t`-values for which points are calculated, and thus influences the smoothness of the graph;[5]

`xMin, xMax, yMin, yMax` specify the range of values on the axes; in our example, `t` is plotted on the horizontal axis, so we used `xMin` = `tMin` and `xMax` = `tMax`;

`xScl, yScl` control placement of the tick marks, just as in ordinary graphing;

`tPlot` determines the `t`-value at which plotting begins; if for example, you were only interested in the part of the solution for $2 \le$ `t` ≤ 5, you could set `tPlot` $= 2$, `xMin` $= 2$, and `xMax` $= 5$ (if you set `tPlot` significantly greater than `tMin`, be patient—even though it only plots from `tPlot` on, it still has to compute all the way from `tMin`);

`difTol` governs accuracy of the computations, and you shouldn't need to change it.

The `[EVAL]` menu choice lets you get the values of the solution for any particular `t`. Use the up and down arrow keys to select the curve you want a value for.

`[AXES]` lets you determine what goes on the horizontal (x) and vertical (y) axes of the graph. The default is to show all the solutions versus `t`; but you can change that. For example, you can plot `Q'1` on the vertical axis and `Q1` on the horizontal (remember to change `xMin`, `xMax`, etc., to some appropriate values).

The SIR model, described in the "Improving on Euler" tutorial, is a good second example on which you might like to try the differential equations module.

[5] The actual graph is made of straight line segments between the calculated points. `tStep` does *not* affect accuracy, only plotting. You would probably get the best looking graph by choosing `tStep` to span exactly one screen pixel: The screen has 128 pixels across, so there are 127 steps from left edge to right edge. Choosing `tStep` = (`xMax`–`xMin`)/127 will result in one step per pixel.

doner@math.ucsb.edu

UCSB Math TI-85 Tutorials:

Graphing DE solutions (SIR model)

Graphing Solutions of Differential Equations: The TI-85 has a built-in module for graphing the (numerical) solutions of differential equations. We could actually write programs that would do this, but the built-in module is faster, more accurate, and more flexible than anything we could easily create. The built-in module uses a numerical method much better than Euler's; unfortunately, its mathematical sophistication precludes discussion of it in this tutorial. A downside is that the built-in module uses a fixed set of variable names, `Q1, Q2,..., Q9`, so you must forego using the names natural to your problem. The big advantage of this module is that you don't have to write programs, and you get both graphical and numeric solutions.

Example: We'll illustrate the use of the module by doing the SIR model (c.f. the description in the "Improving on Euler" tutorial):

$$\begin{aligned} S' &= -.0001SI \\ I' &= .00001SI - I/14 \\ R' &= I/14 \\ S(0) = 45400,\ & I(0) = 2100,\ R(0) = 2500. \end{aligned}$$

This system has no symbolic solution—the typical situation. The first step is to place the calculator in its *differential equation graphing mode*. From the home screen, use the `MODE` command (the secondary function of the `MORE` key), then use the arrow keys to move the cursor to the line

```
Func   Pol   Param   DifEq
```

Move the cursor over to the `DifEq` item and press `ENTER`. Use `EXIT` to return to the home screen. Press `GRAPH`. As you see, the menus are different from those in ordinary graphing mode. Select the menu item `Q'(t)=`. If you see equations already there, use the `[DELf]` menu item repeatedly to delete them. Now type a new set of equations

```
Q'1=-.00001*Q1*Q2
Q'2=.00001*Q1*Q2-Q2/14
Q'3=Q2/14
```

Pressing `ENTER` at the end of a line, or moving the cursor below the last line, causes the calculator to begin a new equation. You don't have to press `ENTER` after the last equation, but it doesn't hurt if you do. As you have probably inferred, we replaced S by `Q1`, I by `Q2`, and R by `Q3`.

Next, you must set the initial conditions. Choose the `[INITC]` menu item, and set up

```
QI1=45400
QI2=2100
QI3=2500
```

You also need to set the ranges—choose the `[RANGE]` item, and set these values:

```
tMin=0
tMax=40
tStep=.5
tPlot=0
xMin=0
xMax=40
xScl=10
yMin=0
yMax=46000
yScl=10000
difTol=.001
```

Now you're ready—choose the `[GRAPH]` menu item and watch it go. When it finishes, you can press `CLEAR` to remove the menu bar from the screen, and `GRAPH` or `EXIT` (once) to bring it back. Try the `[TRACE]` item; you can move the cursor forward from the beginning with the right arrow, and you can use the up and down arrows to move it from one graph to another (the number in the upper right of the screen tells you which curve it's on).

`tMin, tMax` govern the range of `t`-values for which the calculations are made;

`tStep` determines the `t`-values for which points are calculated, and thus influences the smoothness of the graph;[6]

`xMin, xMax, yMin, yMax` specify the range of values on the axes; in our example, `t` is plotted on the horizontal axis, so we used `xMin` = `tMin` and `xMax` = `tMax`;

`xScl, yScl` control placement of the tick marks, just as in ordinary graphing;

`tPlot` determines the `t`-value at which plotting begins; if for example, you were only interested in the part of the solution for $2 \le$ `t` ≤ 5, you could set `tPlot` $= 2$, `xMin` $= 2$, and `xMax` $= 5$ (if you set `tPlot` significantly greater than `tMin`, be patient—even though it only plots from `tPlot` on, it still has to compute all the way from `tMin`);

`difTol` governs accuracy of the computations, and you shouldn't need to change it.

The `[EVAL]` menu choice lets you get the values of the solution for any particular `t`. Use the up and down arrow keys to select the curve you want a value for.

`[AXES]` lets you determine what goes on the horizontal (x) and vertical (y) axes of the graph. The default is to show all the solutions versus `t`; but you can change that. For example, you can plot `Q'1` on the vertical axis and `Q1` on the horizontal (remember to change `xMin`, `xMax`, etc., to some appropriate values).

The sin–cos equations, described in the "Differential Equations" tutorial, is a another good example on which you might like to try the differential equations module.

[6] The actual graph is made of straight line segments between the calculated points. `tStep` does *not* affect accuracy, only plotting. You would probably get the best looking graph by choosing `tStep` to span exactly one screen pixel: The screen has 128 pixels across, so there are 127 steps from left edge to right edge. Choosing `tStep` = (`xMax` – `xMin`)/127 will result in one step per pixel.

doner@math.ucsb.edu