# PPX Exchange

Vol. 6 Number 1 Copyright 1982 January/February 1982

## PROGRAMMING CORNER

*(This column serves a dual purpose. It informs members of what non-PPX software is currently available and also lists descriptions of programs our members would like to see.)*

**TAX Time**

As the dreaded date of April 15 approaches you may want to consider using your TI-59 to aid in the tax calculation process. Listed below is software available for Federal Income Tax calculations from sources other than PPX.

**Cal-Q-Tax**

Cal-Q-Tax from Tax Management Inc. is a series of three Solid State Software™ Modules designed to handle tax computations.

- 1981 Federal Income Tax Module contains individual income tax, lump-sum distribution, tax tables and schedules, Fiduciary income tax, and state and local schedules.
- Estate Tax Module includes family estate planning, marital deduction, estate tax, gift tax, and interrelated deductions.
- Tax Preparation Module includes Schedule G, 1040A, Form 2210, Form 4726, Form 6251, earned income credit, and Federal tax schedules and tables.

For further information contact:

Tax Management Inc.
1231 25th Street N.W.
Washington D.C. 20037

**WG&L Tax Planner™**

Warren, Gorham and Lamont, Inc. has developed a tax planning system using a Solid State Software™ module to calculate and review tax alternatives.

- The income tax program chip calculates regular tax, maximum tax, income averaging, alternative minimum tax, minimum tax, and table tax. Allows the user to test alternatives and change data at any time.
- The estate and gift tax program chip calculates federal estate tax, federal maximum death tax credit, federal gift tax, maximum marital deduction, family estate planning, and New York and California estate and gift taxes.

For further information contact:

Warren, Gorham and Lamont, Inc.
210 South Street
Boston, MA 02111
(800) 225-2363 (Outside Massachusetts)
(617) 423-2020 (Massachusetts residents)

*continued on page 9*

## potpourri

1. In order to allow members to take advantage of the wealth of TI-59 programming hints and other useful information contained in the **past issues** of the PPX Exchange, PPX is now offering these back issues for sale. To facilitate the tapping of these resources the compiling of a **topical index** of these previous newsletters has been completed.

In a change from our previous policy, individual copies of back issues will no longer be available, but the newsletters will be offered in four volumes - one for each of the years 1978-1981. Each of these volumes is available for $7.00 each, and an index will be included with each volume. To order a volume(s) of past newsletters specify the year(s) desired on a PPX order form or plain white paper. Please include the standard $2.00 postage and handling charge for each order plus applicable state tax. (Note: Due to the depletion of original copies of some of the newsletters, we have had to reprint some of these issues. These reprints are on white paper but are of readable quality.) For you old timers who have collected all the back issues, the aforementioned index can be obtained by sending a self addressed, stamped envelope to the Exchange editor.

2. When ordering PPX programs and accessories, please be sure to include the order and the payment in the same envelope. Due to the large volume of orders received, we have been experiencing difficulty in matching orders and payments that arrive separately. We appreciate your cooperation in this area.

---

## Root Finding: A Natural Application

**By Blake DeBerry and Jay Claborn**

There are a multitude of techniques to find roots of equations, and each one has its advantages and disadvantages. It is the aim of this article to examine a few of the more popular methods and demonstrate their implementation on the TI-59. Before we jump into these methods feet first, let's be sure we all know what a root is. A root is defined as a value which reduces an equation to an identity when substituted in for one variable. In other words, it is a value of x that reduces

*continued on page 6*

# Yet Another Look at Alphanumeric Printing of Numerics

**By Jay Claborn**

In the January/February and May/June 1981 issues of the "Exchange" we examined several routines that convert a number into its equivalent alphanumeric codes to allow for printing. The advantage of using alphanumeric codes to print a number over use of the PRT or OP 06 commands is that the use of these codes allows a programmer the flexibility to create a printer output format especially suited to his particular application. Judging from the correspondence I continue to receive on this topic, this technique has received widespread usage. It has never been my intention to "beat a dead dog" (if you will excuse the vulgarism), but as a result of the continued interest in this subject, the techniques have become even more refined. I will attempt to share these refinements with you without replowing the ground previously covered. If you are unfamiliar with the use of numeric-to-alpha routines there is information on how to obtain copies of past newsletters in the "Potpourri" column.

In general, a maximum of five digits per pass is entered into a numeric-to-alpha subroutine to be translated into the corresponding alphanumeric code of up to ten digits. Subroutines that perform this function have been dubbed "5 digit converters", even though they are capable of generating the alpha codes for integers from 0 to 99999. I have further classified these 5 digit converters by the format of their generated alpha codes as routines which print with leading zeroes and without leading zeroes. The formulation of these distinctions may seem quite vagarious until one considers the applications of numeric-to-alpha subroutines. Consider, for instance, the application in which one wishes to generate two columns of integers side-by-side in which the magnitudes of the numbers may vary from one to eight digits. To accomplish this task the first column could be right hand justified in print register 2 (OP 02) with as many as three digits running over into print register 1. Similarily, the second column would be right hand justified in print register 4 and would run over into the right most three positions of print register 3. A sample of the desired type of output might appear as shown below.

OP 01OP 02OP 03OP 04

| | | | |
|---|---|---|---|
| 7500314 | | | 23 |
| | 888 | 32707924 | |

Obviously, since we can only convert five digits at a time to alphanumeric code, the numbers with more than five digits will have to be broken into segments and then translated. By dividing 7500314 in the sample output above by 100000, taking the INV INT of the result, and multiplying by 100000, we have extracted the first five digits (00314) for entry into a 5 digit converter. To extract the remaining digits all we have to do is divide the original number by 100000 and apply the INT function to the result. Now if we were to follow this procedure employing a 5 digit converter that does not produce leading zeroes the output would appear as follows.

| | | | |
|---|---|---|---|
| 75 | 314 | | 23 |
| | 888 | 327 | 7924 |

Because of the omitted zeroes, this attempt is rendered unsatisfactory. We can get better results by using the same converter for print registers 1 and 3 and a separate 5 digit converter that provides leading zeroes to create the codes for print registers 2 and 4. Doing this yields the following.

| | |
|---|---|
| 7500314 | 00023 |
| 00888 | 32707924 |

This is closer to the desired format but still no cigar. By using a little logic we can obtain the desired output.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 063 | 76 | LBL | 076 | 05 | 5 | 089 | 00 | 00 |
| 064 | 99 | PRT | 077 | 22 | INV | 090 | 71 | SBR |
| 065 | 55 | ÷ | 078 | 28 | LOG | 091 | 69 | OP |
| 066 | 05 | 5 | 079 | 49 | PRD | 092 | 92 | RTN |
| 067 | 22 | INV | 080 | 00 | 00 | 093 | 76 | LBL |
| 068 | 28 | LOG | 081 | 00 | 0 | 094 | 98 | ADV |
| 069 | 75 | - | 082 | 32 | X:T | 095 | 43 | RCL |
| 070 | 22 | INV | 083 | 67 | EQ | 096 | 00 | 00 |
| 071 | 59 | INT | 084 | 98 | ADV | 097 | 71 | SBR |
| 072 | 42 | STO | 085 | 71 | SBR | 098 | 68 | NOP |
| 073 | 00 | 00 | 086 | 68 | NOP | 099 | 92 | RTN |
| 074 | 95 | = | 087 | 32 | X:T | | | |
| 075 | 32 | X:T | 088 | 43 | RCL | | | |

If LBL OP contains a 5 digit converter with leading zeroes and LBL NOP is a 5 digit converter without leading zeroes, we are in business. When sending "7500314" into LBL PRT, steps 063-082 result in a "75" in the display, "314" in register 00, and a "0" in the t-register. Steps 083-084 test to see if one or two print registers will be needed to print the number and transfer to LBL ADV if only one print register is required. In our case "75" does not equal "0" therefore two print registers will be required, and the transfer is not made. Steps 085-087 fetch the alpha codes for the most significant three (or less) digits of the number from subroutine NOP and put them in the t-register. Steps 088-092 use subroutine OP to generate the alpha codes for the five least significant digits leaving them in the display register. Upon return from subroutine PRT all that has to be done to load the proper print registers is to perform OP 02 x↔t OP 01.

Now that the need for different types of 5 digit converters has been established, let us turn our attention to the actual 5 digit converter routines. In trying to create an "optimum" 5 digit converter, the variable that is usually optimized is the routine speed. Throughout this exploration of different conversion routines the assumption will be made that run time is indeed the quantity to be minimized. The fastest 5 digit converter that does not produce leading zeroes was submitted by Dick Collins. Collins's routine is a slight revision of Bill Beebe's program that appeared in the May/June 1981 issue. In a test consisting of printing 5 digit numbers ten times, Collins's version had a run time of 66.6 seconds versus 72.9 for the Beebe version. The speed of Collins's routine is highly dependent on its location at the head of a

program due to the use of label addressing. (Yes! Label addressing can be faster than absolute addressing if the label is located early in the program.) Collins's subroutine, appropriately labeled NOP to be consistent with our previous notation, is shown below.

```
000  76 LBL   014  22 INV   028  01  1
001  68 NOP   015  59 INT   029  00  0
002  32 X:T   016  75  -    030  82 HIR
003  01  1    017  53  (    031  48 48
004  82 HIR   018  24 CE    032  95  =
005  08 08    019  85  +    033  77 GE
006  32 X:T   020  93  .    034  77 GE
007  76 LBL   021  01  1    035  65  ×
008  77 GE    022  85  +    036  82 HIR
009  55  ÷    023  59 INT   037  18 18
010  01  1    024  55  ÷    038  33 X²
011  00  0    025  05  5    039  95  =
012  75  -    026  54  )    040  92 RTN
013  53  (    027  55  ÷
```

Dick Collins also created the fastest 5 digit converter with leading zeroes. Again, the algorithm is a variation of Bill Beebe's method. In the same test mentioned previously, this routine took 59.5 seconds. The clever use of label addressing by placing it at the front of program memory again means this routine will run slower if it is relocated further back in the program memory. This routine which starts at LBL OP is listed below.

```
000  76 LBL   013  93  .    026  69 OP
001  10 E'    014  01  1    027  10 E'
002  55  ÷    015  85  +    028  10 E'
003  01  1    016  59 INT   029  10 E'
004  00  0    017  55  ÷    030  10 E'
005  75  -    018  05  5    031  10 E'
006  53  (    019  54  )    032  65  ×
007  22 INV   020  55  ÷    033  05  5
008  59 INT   021  01  1    034  22 INV
009  75  -    022  00  0    035  28 LOG
010  53  (    023  95  =    036  33 X²
011  24 CE    024  92 RTN   037  95  =
012  85  +    025  76 LBL   038  92 RTN
```

To achieve the two column printing that was shown earlier, it would appear that it will require 117 steps (37 for LBL PRT, 42 for LBL NOP, and 39 for LBL OP). As many of you have probably already noted, labels NOP and OP have many steps in common so that the two routines can be integrated to occupy only 63 steps together. This combining process does, however, slightly impede the execution of LBL NOP.

```
000  76 LBL   021  01  1    042  95  =
001  10 E'    022  00  0    043  92 RTN
002  55  ÷    023  92 RTN   044  76 LBL
003  01  1    024  76 LBL   045  69 OP
004  00  0    025  68 NOP   046  10 E'
005  75  -    026  32 X:T   047  95  =
006  53  (    027  01  1    048  10 E'
007  22 INV   028  82 HIR   049  95  =
008  59 INT   029  08 08    050  10 E'
009  75  -    030  32 X:T   051  95  =
010  53  (    031  10 E'    052  10 E'
011  24 CE    032  82 HIR   053  95  =
012  85  +    033  48 48    054  10 E'
013  93  .    034  95  =    055  95  =
014  01  1    035  77 GE    056  65  ×
015  85  +    036  00 00    057  05  5
016  59 INT   037  31 31    058  22 INV
017  55  ÷    038  65  ×    059  28 LOG
018  05  5    039  82 HIR   060  33 X²
019  54  )    040  18 18    061  95  =
020  55  ÷    041  33 X²    062  92 RTN
```

# A CHALLENGE

For those of you who have found this information on the printing of numerics remedial and for those of you that just enjoy a programming challenge (judging from the response to the last programming contest there are quite a few in this category), I have devised a useful application of numeric printing for you to tackle. The task is to create a program that will list all thirteen digits of the contents of data registers 00 through 89 (leaving the first bank of memory for the program and any working registers needed) in the format shown below.

```
00 3.141592653590 00
01-2.718281828459-13
02 0.000000000000 00
03 2.718281828459 26
```

The register number is printed on the left followed by the thirteen digit mantissa and the power of ten by which it is multiplied. As shown, the program should be able to list all possible register contents in scientific format including both positive and negative numbers. In order to aid in the judging process all entries must meet the following criteria.

1) The program should be able to list the contents of registers 0-89 leaving the contents of these registers intact.
2) The program should be initiated by entering the register number of the first register to be listed and pressing A.
3) The program should run as fast as possible. Program speed will be the only judging criterion assuming the other requirements are met. (All submissions will be timed on the same TI-59/PC-100A combination to avoid run time differences due to different machines.) Since this is a contest of programming skill and technique, no fast mode programs will be considered.
4) All programs should be submitted by March 31, 1982 with a PC-100 tape listing, pre-recorded magnetic card (recorded in power-up partition), and an estimate of the time required to list the contents of ten registers. Regular PPX submission forms need not be used, and submitted magnetic cards will not be returned.

The fastest program will be featured in the May/June issue, and the winning author will receive two magnetic card cases with cards. In order to give you a time to beat, I have already written such a program. My program turns in a sluggish time of 3 minutes and 41 seconds for ten registers; surely you can beat that!

The PPX Exchange is published bimonthly and is the only newsletter published by Texas Instruments for TI-59 owners. Members are invited to contribute articles and items of general interest to other TI-59 users. Authors of accepted feature articles for the newsletter will receive their choice of either a one year complimentary PPX membership or a Solid State Software™ module. Please double-space and type all submissions, and forward them to:

Texas Instruments, PPX
P.O. Box 53
Lubbock, Texas 79408
Attn: PPX Exchange Editor

# from the Analyst's Desk

•Since the publication of Robert Wyer's "Clearing Your Memory" in the March/April 1981 PPX Exchange a couple of other techniques to clear blocks of data registers have surfaced. The first routine which was contributed by PPX member Marcelo Falcon is an excellent application of the Master Library module. He relates that all that is necessary to clear the block of registers from 1 to N (where N is any register number greater than zero allowed within the partitioning) is to perform the sequence: N Pgm 01 SBR 012. To understand how this routine works one can download ML-01 and examine program steps 012 through 021.

| | | |
|---|---|---|
| 012 | 42 | STO |
| 013 | 01 | 01 |
| 014 | 00 | 0 |
| 015 | 72 | ST* |
| 016 | 01 | 01 |
| 017 | 97 | DSZ |
| 018 | 01 | 01 |
| 019 | 00 | 00 |
| 020 | 15 | 15 |
| 021 | 92 | RTN |

These steps are actually part of subroutine CLR which is used to initialize the calculator for statistics and linear regression. The advantage of this routine is that since it is a module library subroutine, it runs faster than its main memory counterpart.

A second routine allows for the clearing of an arbitrary, user-defined block of data registers. This routine can clear any block of registers within the partition as long as register 0 is not included; register 0 and the t-register are used by the routine. To use the routine place the number of the lowest register to be cleared in the t-register and the highest register to be cleared in the display and call SBR CLR.

| | | | | | |
|---|---|---|---|---|---|
| 000 | 76 | LBL | 008 | 30 | 30 |
| 001 | 25 | CLR | 009 | 43 | RCL |
| 002 | 42 | STO | 010 | 00 | 00 |
| 003 | 00 | 00 | 011 | 77 | GE |
| 004 | 00 | 0 | 012 | 00 | 00 |
| 005 | 72 | ST* | 013 | 04 | 04 |
| 006 | 00 | 00 | 014 | 92 | RTN |
| 007 | 69 | DP | | | |

•Marcelo Falcon shares the following concerning the accessing of the hierarchy registers from the keyboard. When developing a program, especially one that makes use of the HIR instruction, one may want to check the contents of a hierarchy register. To accomplish this simply enter the following sequence anywhere in program memory: LBL GTO HIR (be sure there is no other LBL GTO in the program). Now, to execute a HIR instruction from the keyboard, press GTO GTO SST XY (where XY is the desired two digit hierarchy function code).

•PPX member Charles Gaylord has informed us of some precise tests to assure that an INV Write command is placed in a "safe" location. As mentioned in the article "Multiple Card Usage" (September/October 1981), the sequence "N INV Write GTO nnn" (or "N INV Write GTO n") must be carefully positioned within a program if the user intends to read new code into the bank through which the program is currently executing. One can successfully read new code into the current bank and go to any available location if the "Write" command is positioned according to one of the following rules.

1) If the sequence is "N INV Write GTO nnn"", position the command "Write" on a step that, when divided by 8, yields a decimal portion from .0 to .5, inclusive.
2) If the sequence is "N INV Write GTO n" or "N INV Write GTO IND xx", position the command "Write" on a step that, when divided by 8, yields a decimal portion from .9 to .625, inclusive.

These rules assure you that your intended address is stored safely in the processing buffer before you overwrite the bank in which the program is currently running. There is one exception; "Write" cannot be positioned on step 000 because that would split the "INV Write" instruction.

•An extension to the "Inverse Days Between Dates" program which appeared in the July/August 1981 issue has been recommended by Henk Leidekker. He suggests the adding of a printing feature which will print not only the date but also the date of the week. Making use of the day of the week feature of Master Library Program 20, this extension can be accomplished. Program and data register listings are shown below. The input sequence remains unchanged. (Enter the starting date in MMDD.YYYY format and press A. Then enter the number of days from the starting date and press C.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 000 | 76 | LBL | 043 | 55 | ÷ | 086 | 92 | 92 |
| 001 | 10 | E' | 044 | 03 | 3 | 087 | 69 | DP |
| 002 | 42 | STO | 045 | 06 | 6 | 088 | 31 | 31 |
| 003 | 00 | 00 | 046 | 05 | 5 | 089 | 71 | SBR |
| 004 | 36 | PGM | 047 | 93 | . | 090 | 01 | 01 |
| 005 | 20 | 20 | 048 | 02 | 2 | 091 | 13 | 13 |
| 006 | 14 | D | 049 | 04 | 4 | 092 | 50 | I×I |
| 007 | 22 | INV | 050 | 95 | = | 093 | 85 | + |
| 008 | 58 | FIX | 051 | 59 | INT | 094 | 01 | 1 |
| 009 | 85 | + | 052 | 42 | STO | 095 | 85 | + |
| 010 | 09 | 9 | 053 | 09 | 09 | 096 | 43 | RCL |
| 011 | 03 | 3 | 054 | 01 | 1 | 097 | 01 | 01 |
| 012 | 95 | = | 055 | 42 | STO | 098 | 65 | × |
| 013 | 42 | STO | 056 | 01 | 01 | 099 | 01 | 1 |
| 014 | 06 | 06 | 057 | 42 | STO | 100 | 00 | 0 |
| 015 | 01 | 1 | 058 | 02 | 02 | 101 | 00 | 0 |
| 016 | 00 | 0 | 059 | 71 | SBR | 102 | 85 | + |
| 017 | 69 | DP | 060 | 01 | 01 | 103 | 43 | RCL |
| 018 | 17 | 17 | 061 | 13 | 13 | 104 | 09 | 09 |
| 019 | 73 | RC* | 062 | 77 | GE | 105 | 55 | ÷ |
| 020 | 06 | 06 | 063 | 00 | 00 | 106 | 04 | 4 |
| 021 | 69 | DP | 064 | 70 | 70 | 107 | 22 | INV |
| 022 | 04 | 04 | 065 | 69 | DP | 108 | 28 | LOG |
| 023 | 43 | RCL | 066 | 39 | 39 | 109 | 95 | = |
| 024 | 00 | 00 | 067 | 71 | SBR | 110 | 10 | E' |
| 025 | 69 | DP | 068 | 01 | 01 | 111 | 92 | RTN |
| 026 | 06 | 06 | 069 | 13 | 13 | 112 | 43 | RCL |
| 027 | 92 | RTN | 070 | 32 | X:T | 113 | 09 | 09 |
| 028 | 76 | LBL | 071 | 50 | I×I | 114 | 42 | STO |
| 029 | 11 | A | 072 | 55 | ÷ | 115 | 03 | 03 |
| 030 | 10 | E' | 073 | 02 | 2 | 116 | 36 | PGM |
| 031 | 36 | PGM | 074 | 09 | 9 | 117 | 20 | 20 |
| 032 | 20 | 20 | 075 | 85 | + | 118 | 71 | SBR |
| 033 | 11 | A | 076 | 01 | 1 | 119 | 00 | 00 |
| 034 | 92 | RTN | 077 | 95 | = | 120 | 86 | 86 |
| 035 | 76 | LBL | 078 | 59 | INT | 121 | 75 | - |
| 036 | 13 | C | 079 | 42 | STO | 122 | 43 | RCL |
| 037 | 85 | + | 080 | 01 | 01 | 123 | 05 | 05 |
| 038 | 43 | RCL | 081 | 71 | SBR | 124 | 95 | = |
| 039 | 04 | 04 | 082 | 01 | 01 | 125 | 32 | X:T |
| 040 | 95 | = | 083 | 13 | 13 | 126 | 00 | 0 |
| 041 | 42 | STO | 084 | 77 | GE | 127 | 92 | RTN |
| 042 | 05 | 05 | 085 | 00 | 00 | | | |

| Code | Register |
|---|---|
| 3613. | 93 |
| 3641. | 94 |
| 3032. | 95 |
| 3741. | 96 |
| 4317. | 97 |
| 3723. | 98 |
| 2135. | 99 |

- PPX member Mark Miller has recommended a shortcut for keying Prt and Adv into programs. When the TI-59 is attached to the print cradle, Prt and Adv may be entered in the LRN mode by simply pressing these keys on the print cradle. This procedure eliminates the need to press the 2nd key on the TI-59 and is helpful when entering several Adv's at a time.
- Notice to members who have purchased the program "Othello" (PPX #918229). This program has been revised. To receive revision B, please return your original copy to PPX.
- In many program submissions we are encountering the instructions to use INV Write from the keyboard to manually read a magnetic card. We would like to caution all users in the use of this instruction from the keyboard. The TI-59 recognizes N INV Write as a valid entry only when used as a program instruction. The use of this key sequence as a means of initially reading a magnetic card can possibly alter or erase the magnetic card. With a zero or the bank number in the display the calculator automatically reads a card placed in the read/write slot provided the partitioning of the TI-59 is the same as the recorded program.

# Alpha Register Lister

**By Bill Beebe**

This program can be very useful in debugging and documenting programs which use prestored alphanumeric data. Using the PC-100A/C this program lists the number stored in the data register, the register number, and the alpha equivalent of the register contents all on a single line for any block of registers from 1 to 89. It will not list any registers which contain zero. (Note: Since this program is designed to list alphanumeric code, it does not properly print the numeric part of numbers greater than 9999999999 or negative numbers, nor does it print the fractional part of a number.) To use the program simply enter the lowest (LL) and highest (HH) registers to be listed in the LL.HH format and press A. The maximum run time for each register is about 18 seconds.

**SAMPLE OUTPUT**

As an example of the use of this program, enter the following alpha codes in their respective registers.

| Alpha Code | Register |
|---|---|
| 32231700 | 01 |
| 3724200612 | 03 |
| 2235171337 | 04 |
| 2141310073 | 05 |

Now enter 1.10 in the display and press A. The program output is shown at the top of the next column.

```
  37231700    1  THE
3724200612    2 TI-59
  24360000    3  IS
2235171337    4 GREAT
2141310073    5 FUN !
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 000 | 76 | LBL | 045 | 75 | - | 090 | 01 | 01 |
| 001 | 16 | A' | 046 | 59 | INT | 091 | 01 | 01 |
| 002 | 32 | X↕T | 047 | 42 | STO | 092 | 11 | 11 |
| 003 | 01 | 1 | 048 | 00 | 00 | 093 | 85 | + |
| 004 | 82 | HIR | 049 | 95 | = | 094 | 82 | HIR |
| 005 | 03 | 03 | 050 | 88 | DMS | 095 | 13 | 13 |
| 006 | 00 | 0 | 051 | 82 | HIR | 096 | 33 | X² |
| 007 | 32 | X↕T | 052 | 18 | 18 | 097 | 65 | × |
| 008 | 95 | = | 053 | 82 | HIR | 098 | 53 | ( |
| 009 | 55 | ÷ | 054 | 04 | 04 | 099 | 35 | 1/X |
| 010 | 01 | 1 | 055 | 73 | RC* | 100 | 65 | × |
| 011 | 00 | 0 | 056 | 00 | 00 | 101 | 01 | 1 |
| 012 | 82 | HIR | 057 | 29 | CP | 102 | 00 | 0 |
| 013 | 43 | 43 | 058 | 69 | DP | 103 | 22 | INV |
| 014 | 75 | - | 059 | 04 | 04 | 104 | 28 | LOG |
| 015 | 59 | INT | 060 | 67 | EQ | 105 | 55 | ÷ |
| 016 | 82 | HIR | 061 | 01 | 01 | 106 | 09 | 9 |
| 017 | 07 | 07 | 062 | 24 | 24 | 107 | 09 | 9 |
| 018 | 85 | + | 063 | 55 | ÷ | 108 | 54 | ) |
| 019 | 93 | . | 064 | 05 | 5 | 109 | 59 | INT |
| 020 | 01 | 1 | 065 | 22 | INV | 110 | 95 | = |
| 021 | 85 | + | 066 | 28 | LOG | 111 | 69 | DP |
| 022 | 59 | INT | 067 | 75 | - | 112 | 02 | 02 |
| 023 | 55 | ÷ | 068 | 22 | INV | 113 | 43 | RCL |
| 024 | 05 | 5 | 069 | 59 | INT | 114 | 00 | 00 |
| 025 | 95 | = | 070 | 82 | HIR | 115 | 16 | A' |
| 026 | 55 | ÷ | 071 | 06 | 06 | 116 | 52 | EE |
| 027 | 01 | 1 | 072 | 95 | = | 117 | 02 | 2 |
| 028 | 00 | 0 | 073 | 67 | EQ | 118 | 22 | INV |
| 029 | 85 | + | 074 | 00 | 00 | 119 | 52 | EE |
| 030 | 82 | HIR | 075 | 78 | 78 | 120 | 69 | DP |
| 031 | 17 | 17 | 076 | 16 | A' | 121 | 03 | 03 |
| 032 | 22 | INV | 077 | 22 | INV | 122 | 69 | DP |
| 033 | 67 | EQ | 078 | 86 | STF | 123 | 05 | 05 |
| 034 | 00 | 00 | 079 | 01 | 01 | 124 | 69 | DP |
| 035 | 08 | 08 | 080 | 69 | DP | 125 | 20 | 20 |
| 036 | 95 | = | 081 | 01 | 01 | 126 | 43 | RCL |
| 037 | 65 | × | 082 | 05 | 5 | 127 | 00 | 00 |
| 038 | 82 | HIR | 083 | 22 | INV | 128 | 32 | X↕T |
| 039 | 13 | 13 | 084 | 28 | LOG | 129 | 82 | HIR |
| 040 | 33 | X² | 085 | 65 | × | 130 | 14 | 14 |
| 041 | 95 | = | 086 | 82 | HIR | 131 | 77 | GE |
| 042 | 92 | RTN | 087 | 16 | 16 | 132 | 00 | 00 |
| 043 | 76 | LBL | 088 | 16 | A' | 133 | 55 | 55 |
| 044 | 11 | A | 089 | 87 | IFF | 134 | 92 | RTN |

**Check Sum For Program**

```
BANK?
          1.
INSERT CARD
        120.
        104.
         80.
PRESS LRN, INS(4x)
      LRN,     C
        124.
         84.
         52.
 100.8499535
```

## ADDRESS CHANGES

In order to ensure uninterrupted service, please submit address changes to PPX at least six weeks prior to the effective date of the change. Send your name, membership number, old and new addresses to:

PPX
P.O. Box 53
Lubbock, TX 79408

## Root Finding (continued from page 1)

some function of x which we will call f(x) so that f(x) = 0. Many equations have more than one root, polynomials being a good example. In general, these roots can be either real or imaginary. For the sake of simplicity, we will only address the subject of finding real roots of a real function.

Because of the involvement or impossibility of algebraically solving for the roots of many questions, numerical methods have been developed which allow us to take advantage of the number crunching and iterative capabilities of computing machines such as the TI-59. As is indicated in its definition, an iterative process (which all numerical root finding techniques are) involves the replication of a cycle of operations to (hopefully) produce results which approximate the desired result closer and closer. Since such techniques do not solve for the root explicitly, a condition must be defined which, when satisfied, will indicate that the iterative process has converged "close enough" to the root for the process to be halted. Such criteria usually require the user to input a small, positive number called epsilon ($\epsilon$). One method of testing for convergence (Type 1) is to check the magnitude of the change in the independent variable from one iteration to the next, i.e. a function of x is given by f(x), and we wish to find a root such that f(x) = 0. If $|x_n - x_{n-1}| < \epsilon$ (where $x_n$ denotes the value of x that approximates the root after the nth iteration), then the value of $x_n$ is considered to be "close enough" to the root to be called a root. A second method of testing for convergence (Type 2) is to test after each iteration to see if $f(x) < \epsilon$. Both of these commonly used methods have their shortcomings. With the first method, it is possible that $|x_n - x_{n-1}|$ may be very small at a location that is not within $\epsilon$ of the root. Such a situation might occur when the function is shaped as shown in figure 1.

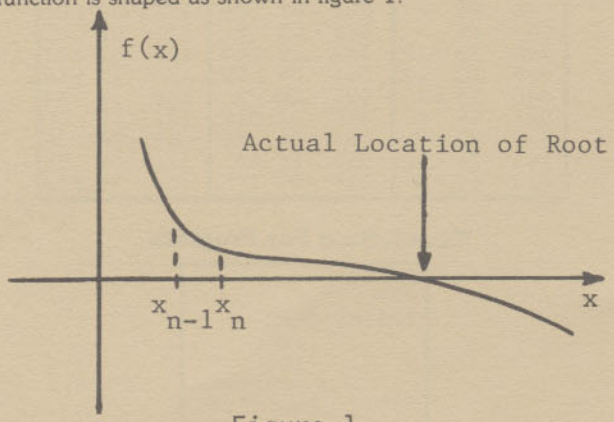


Figure 1

Example of Failure of Type 1 Convergence Testing

The second criterion can fail when the function is very flat in the region surrounding the root. In such a case, f(x) is very small, but x can be further than $\epsilon$ away from the root. If there is a great need to minimize these errors, we recommend a third convergence criterion (Type 3); $[x_n - x_{n-1})^2 + f(x_{n-1})^2]^{1/2} < \epsilon$. This criterion tests the length of the segment shown in figure 2.

Now that we have determined how to tell when a root has been found, let's examine five of the most commonly used iteration techniques.

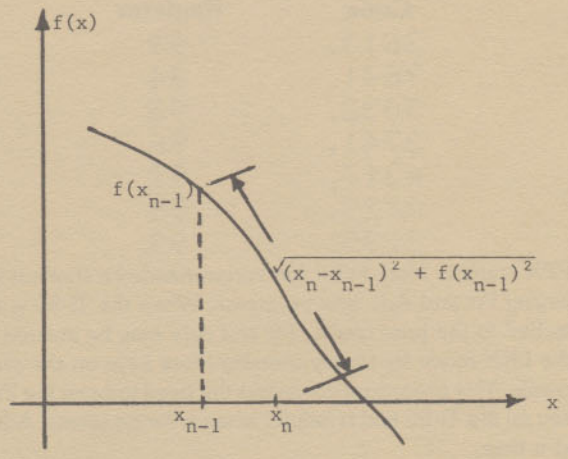


Figure 2

Type 3 Convergence Testing

### Simple Iteration

One of the easiest methods to implement is that of simple iteration. If the function f(x) = 0 can be rearranged such that x = g(x), then the iterative process $x_{n+1} = g(x_n)$ will sometimes yield a root. For this procedure to be successful, the absolute value of the first derivative of g(x) with respect to x must be less than one when evaluated at a root. Because of the form of this process, the most commonly used test of convergence is of the first type discussed above. The flowchart shown in figure 3 illustrates this process.



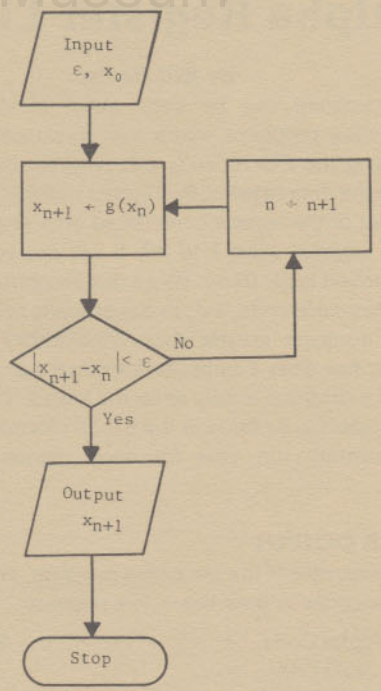


FIGURE 3

FLOWCHART FOR SIMPLE ITERATION

As an example consider the cubic equation $x^3 + 5x^2 - 64x - 140 = 0$ which has roots at 7, -2, and -10. Rearranging, we can write the equation as $x = 140/(x^2 + 5x - 64)$ which is of the form $x = g(x)$.

This method is easily coded on the TI-59. In an effort to make the routine as general as possible we will use LBL A' to calculate g(x). While LBL A' will have to be changed for every different function, the rest of the program can remain the same.

To use the program (listed in figure 4), enter an initial guess for the value of the root and press A. Enter the value of $\epsilon$ and press E (if this step is not performed a default value of .01 will be used for $\epsilon$). Press C to start the calculations. The calculated value for the root will be displayed. With g(x) defined as it is, the program will find the root at -2 since the absolute value of the derivative of g(x) is greater than 1 at x = -10 and x = 7.

| | | | | | |
|---|---|---|---|---|---|
| 000 | 76 | LBL | 026 | 76 | LBL |
| 001 | 11 | A | 027 | 16 | A' |
| 002 | 42 | STO | 028 | 43 | RCL |
| 003 | 01 | 01 | 029 | 01 | 01 |
| 004 | 93 | . | 030 | 33 | X² |
| 005 | 00 | 0 | 031 | 85 | + |
| 006 | 01 | 1 | 032 | 05 | 5 |
| 007 | 32 | X↕T | 033 | 65 | × |
| 008 | 91 | R/S | 034 | 43 | RCL |
| 009 | 76 | LBL | 035 | 01 | 01 |
| 010 | 15 | E | 036 | 75 | - |
| 011 | 32 | X↕T | 037 | 06 | 6 |
| 012 | 91 | R/S | 038 | 04 | 4 |
| 013 | 76 | LBL | 039 | 95 | = |
| 014 | 13 | C | 040 | 35 | 1/X |
| 015 | 16 | A' | 041 | 65 | × |
| 016 | 75 | - | 042 | 01 | 1 |
| 017 | 48 | EXC | 043 | 04 | 4 |
| 018 | 01 | 01 | 044 | 00 | 0 |
| 019 | 95 | = | 045 | 95 | = |
| 020 | 50 | I×I | 046 | 92 | RTN |
| 021 | 77 | GE | | | |
| 022 | 13 | C | | | |
| 023 | 43 | RCL | | | |
| 024 | 01 | 01 | | | |
| 025 | 91 | R/S | | | |

Figure 4
Listing for Simple
Iteration Program

**Bisection**

Bisection is one of the easiest methods to visualize; however, due its "brute force" nature, it tends to require more iterations to converge than the more elegant methods. In order to visualize how this technique works, consider the function in figure 5 which has a single real root bounded by two points, $x_L$ and $x_R$, on the x-axis. The points $x_L$ and $x_R$ must be such that $f(x_L) \cdot f(x_R) < 0$, or, in words, the function must be on opposite sides of the x-axis at the interval boundaries, $x_L$ and $x_R$. To find the root, we first bisect the interval by calculating its midpoint $x_m = (x_L + x_R)/2$. Next, we compute the product $f(x_m) \cdot f(x_R)$. If this product is negative then the root is between $x_m$ and $x_R$, and we let $x_m$ become our new left end bound ($x_L$). However, if the product is positive then the root is between $x_L$ and $x_m$, and we let $x_m$ become our new right end bound. This process is continued until the root has been found as accurately as desired. Since the maximum error in each $x_m$ is $\frac{1}{2}(x_L - x_R)$, the process is usually considered convergent when $x_R - x_L < 2\epsilon$.

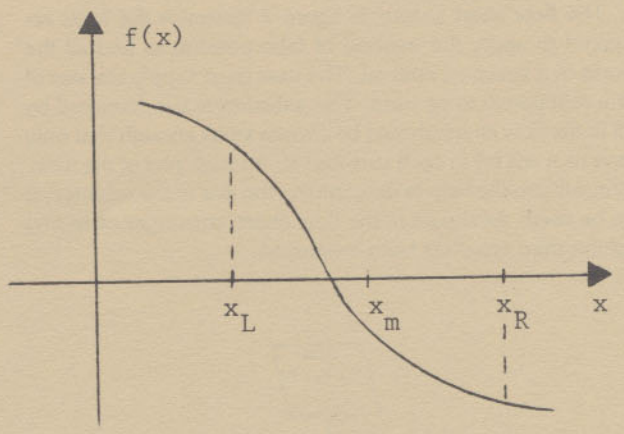


Figure 5
Illustration of Bisection
Method

Although bisection is a "crude" method, it will always find the root if it is supplied with two endpoints as described above. If one wishes to find more than just one root in an interval, The complexity of the algorithm is greatly increased. Since we will take an indepth look at finding more than one root in our consideration of the next method, we will leave it to the interested reader to download Master Library Program 08 for an example of the implementation of bisection to find more than one root.

**Regula Falsi** (false-position)

The method of false position is one step up the ladder of sophistication in root finding from the method of bisection. This method is often used in preference to bisection because it utilizes the same information as bisection but usually generates a closer approximation of the root. As shown in figure 6, the approximation of the root (call it $x_N$) is the point where the line defined as the two points $(x_L, f(x_L))$ and $(x_R, f(x_R))$ crosses the x-axis. Once the approximation $x_N$ has been found by

$$x_N = x_L + (x_R - x_L)\, f(x_L) / [f(x_L) - f(x_R)]$$

the procedure for determining which side of $x_N$ the actual root lies is the same as that for bisection.

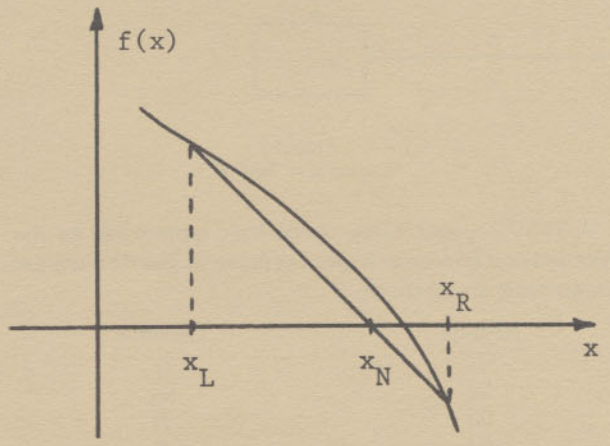


Figure 6
Illustration of Regula Falsi
Method

The flow chart shown in figure 7 illustrates the logic required to apply the method of false-position to find all the roots in a specified interval. The user must supply the size of the subinterval to be used. This subinterval size (denoted by N in the flow chart) should be chosen small enough that only one root will fall in each subinterval. A rough plot of the function will usually help in determining the size of the subinterval to be used. As shown in the flow chart, convergence testing of the third type has been employed.

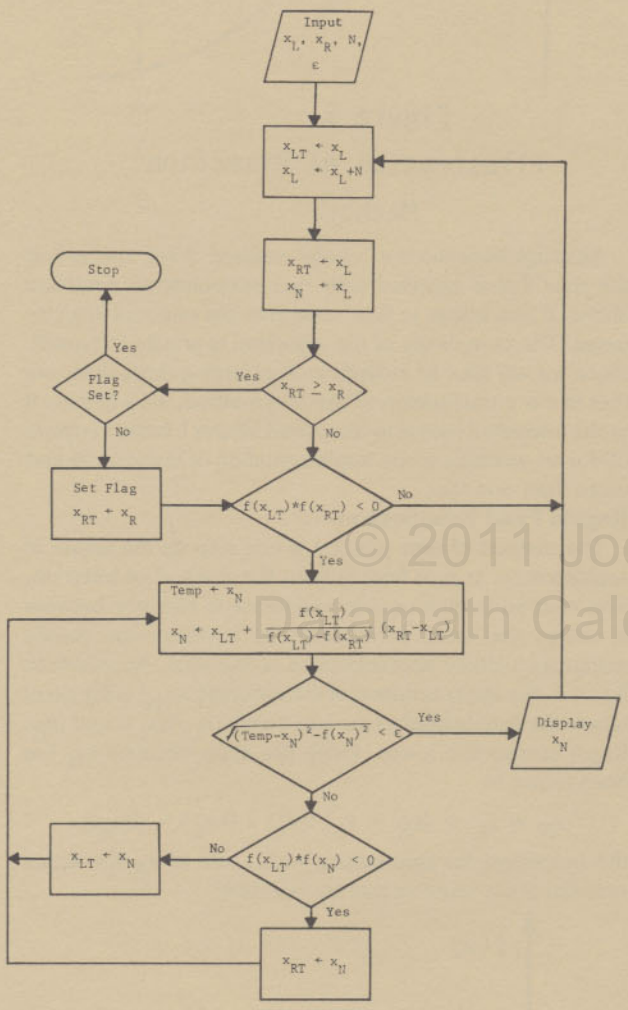


FIGURE 7
REGULA FALSI METHOD

A TI-59 keycode listing of the logic represented by the false-position flow chart is given in figure 8. The data register assignments are as shown.

| Register | Contents |
|---|---|
| 01 | $x_L$ |
| 02 | $x_R$ |
| 03 | N |
| 04 | E |
| 05 | $x_{LT}$ |
| 06 | $x_{RT}$ |
| 07 | $x_N$ |
| 08 | $(x_N - \text{Temp})^2$ |
| 09 | $f(x_{LT})$ |
| 10 | $f(x_{RT})$ |
| 11 | $f(x_N)$ |
| 12 and above | Available for use by subroutine A' |

Subroutine A' is reserved for evaluation of the function of interest. The sample cubic previously considered is coded in subroutine A' of figure 8. To find the three roots of the sample problem (-10, -2, and 7) enter a left bound (say -20) and press A. Enter a right bound (10) and press B. Select a suitable subinterval size (7) and press C. Enter a small positive number for ε and press E (if a value for ε is not entered, a default value of 0.01 is assigned). The root at -10 will be found first and displayed. Press R/S to continue the program. After all roots in the interval have been displayed, the display will flash.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | 76 | LBL | 045 | 42 | STO | 090 | 06 | 06 | 135 | 06 | 06 |
| 001 | 11 | A | 046 | 07 | 07 | 091 | 75 | - | 136 | 43 | RCL |
| 002 | 42 | STO | 047 | 32 | X:T | 092 | 43 | RCL | 137 | 11 | 11 |
| 003 | 01 | 01 | 048 | 43 | RCL | 093 | 05 | 05 | 138 | 42 | STO |
| 004 | 22 | INV | 049 | 02 | 02 | 094 | 54 | ) | 139 | 10 | 10 |
| 005 | 86 | STF | 050 | 77 | GE | 095 | 85 | + | 140 | 61 | GTO |
| 006 | 01 | 01 | 051 | 00 | 00 | 096 | 43 | RCL | 141 | 00 | 00 |
| 007 | 93 | . | 052 | 61 | 61 | 097 | 05 | 05 | 142 | 78 | 78 |
| 008 | 00 | 0 | 053 | 87 | IFF | 098 | 75 | - | 143 | 43 | RCL |
| 009 | 01 | 1 | 054 | 01 | 01 | 099 | 48 | EXC | 144 | 07 | 07 |
| 010 | 42 | STO | 055 | 01 | 01 | 100 | 07 | 07 | 145 | 42 | STO |
| 011 | 04 | 04 | 056 | 54 | 54 | 101 | 95 | = | 146 | 05 | 05 |
| 012 | 91 | R/S | 057 | 86 | STF | 102 | 33 | X² | 147 | 43 | RCL |
| 013 | 76 | LBL | 058 | 01 | 01 | 103 | 42 | STO | 148 | 11 | 11 |
| 014 | 12 | B | 059 | 42 | STO | 104 | 08 | 08 | 149 | 42 | STO |
| 015 | 42 | STO | 060 | 06 | 06 | 105 | 43 | RCL | 150 | 09 | 09 |
| 016 | 02 | 02 | 061 | 43 | RCL | 106 | 07 | 07 | 151 | 61 | GTO |
| 017 | 91 | R/S | 062 | 06 | 06 | 107 | 16 | A' | 152 | 00 | 00 |
| 018 | 76 | LBL | 063 | 16 | A' | 108 | 42 | STO | 153 | 78 | 78 |
| 019 | 15 | E | 064 | 42 | STO | 109 | 11 | 11 | 154 | 25 | CLR |
| 020 | 42 | STO | 065 | 10 | 10 | 110 | 33 | X² | 155 | 35 | 1/X |
| 021 | 04 | 04 | 066 | 43 | RCL | 111 | 85 | + | 156 | 91 | R/S |
| 022 | 91 | R/S | 067 | 05 | 05 | 112 | 43 | RCL | 157 | 76 | LBL |
| 023 | 76 | LBL | 068 | 16 | A' | 113 | 08 | 08 | 158 | 16 | A' |
| 024 | 13 | C | 069 | 42 | STO | 114 | 95 | = | 159 | 42 | STO |
| 025 | 42 | STO | 070 | 09 | 09 | 115 | 34 | fX | 160 | 12 | 12 |
| 026 | 03 | 03 | 071 | 65 | × | 116 | 32 | X:T | 161 | 65 | × |
| 027 | 91 | R/S | 072 | 43 | RCL | 117 | 43 | RCL | 162 | 53 | ( |
| 028 | 43 | RCL | 073 | 10 | 10 | 118 | 04 | 04 | 163 | 24 | CE |
| 029 | 07 | 07 | 074 | 95 | = | 119 | 77 | GE | 164 | 33 | X² |
| 030 | 91 | R/S | 075 | 29 | CP | 120 | 00 | 00 | 165 | 85 | + |
| 031 | 76 | LBL | 076 | 77 | GE | 121 | 28 | 28 | 166 | 05 | 5 |
| 032 | 14 | D | 077 | 14 | D | 122 | 43 | RCL | 167 | 65 | × |
| 033 | 43 | RCL | 078 | 43 | RCL | 123 | 11 | 11 | 168 | 43 | RCL |
| 034 | 01 | 01 | 079 | 09 | 09 | 124 | 65 | × | 169 | 12 | 12 |
| 035 | 42 | STO | 080 | 55 | ÷ | 125 | 43 | RCL | 170 | 75 | - |
| 036 | 05 | 05 | 081 | 53 | ( | 126 | 09 | 09 | 171 | 06 | 6 |
| 037 | 85 | + | 082 | 24 | CE | 127 | 95 | = | 172 | 04 | 4 |
| 038 | 43 | RCL | 083 | 75 | - | 128 | 29 | CP | 173 | 54 | ) |
| 039 | 03 | 03 | 084 | 43 | RCL | 129 | 77 | GE | 174 | 75 | - |
| 040 | 95 | = | 085 | 10 | 10 | 130 | 01 | 01 | 175 | 01 | 1 |
| 041 | 42 | STO | 086 | 54 | ) | 131 | 43 | 43 | 176 | 04 | 4 |
| 042 | 01 | 01 | 087 | 65 | × | 132 | 43 | RCL | 177 | 00 | 0 |
| 043 | 42 | STO | 088 | 53 | ( | 133 | 07 | 07 | 178 | 95 | = |
| 044 | 06 | 06 | 089 | 43 | RCL | 134 | 42 | STO | 179 | 92 | RTN |

Figure 8
Listing of Regula
Falsi Program

**Newton's Method**

Sir Isaac Newton recommended that the line used in the method of false position be replaced by a line tangent to the function at the current approximation of the root. This method of root finding does not require that the root be bounded, only that an initial approximation in the vicinity of the root be supplied.

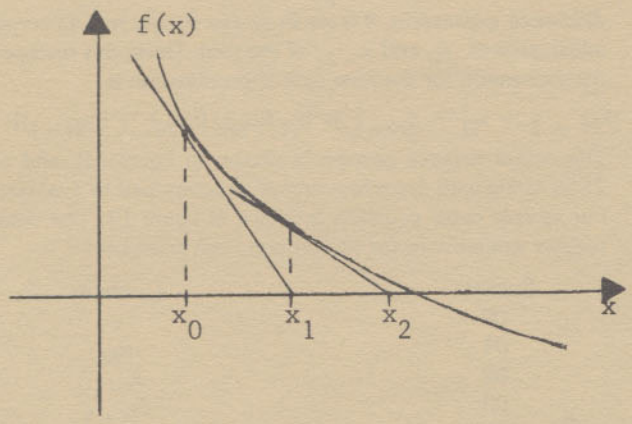


Figure 9
Illustration of Newton's Method

In order to find the location of the new approximation ($x_{N+1}$) the tangent line is defined by the point ($x_N$, $f(x_N)$) and the slope of the line tangent to the function at that point. This slope is known in differential calculus as the first derivative of the function at $x_N$ and is denoted by $f'(x_N)$. Using these notations the next approximation of the root is given by

$$x_{N+1} = x_N - f(x_N)/f'(x_N).$$

This process is depicted in the flow chart of figure 10. Not only does Newton's method tend to converge in fewer iterations than false position, it can also find a root where the function does not cross the x-axis but is tangent to the x-axis. Without large modifications in the logic, Newton's method will only find one root for each initial guess. To find all the roots of a polynomial by Newton's method, synthetic division can be used to find the derivative and to reduce the polynomial once a root is located. It is beyond the scope of this article to fully delve into the application of synthetic division; however, reference 3 listed at the end of this article contains an excellent treatment of the subject.

*(continued on page 10)*

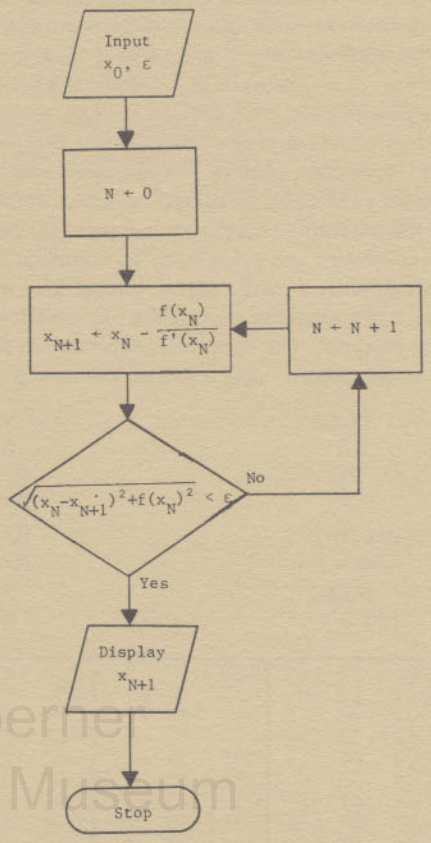


FIGURE 10
NEWTON'S METHOD



## Programming Corner (continued from page 1)

**NEW BOOK ANNOUNCEMENT**

**Using Programmable Calculators for Business**

C. Louis Hohenstein

Designed to help users be able to immediately put to use the power of the TI-59 to solve numerous business problems. Provides fully working programs for payroll calculation, depreciation, tax computations, invoice extensions, forecasting, real rate of return calculations and a host of other business applications.

Available from: Delta Business Publications
P.O. Drawer 166
1175 Peachtree St. NE
Atlanta, GA 30361

**PROGRAMS WANTED**

In order to help PPX members obtain the software they need, we publish program requests. Members who respond to these requests by submitting a PPX program are rewarded with special incentives. All such submissions should be on standard PPX submission forms. The author of the program found to be most suitable to fill each request will receive a Solid State Software™ module of their choice. Runners-up will receive a Specialty Pakette. When submitting a program to fill a "Programming Corner" request, please attach a note stating which request the submission is intended to fill.

The program requests for this issue are listed below. All submissions to fill these requests should be postmarked no later than April 30, 1982.

- A program for psychometric analysis for air conditioning given relative humidity, dry bulb temperature, and elevation above sea level. Output required: water-air ratio, specific volume, percent saturation, and enthalpy of the mixture.
- A program to compute the $Y^X$ function for large numbers with greater than 10 digit arguments.
- A program to analyze and forecast time series data which can contain horizontal, seasonal, cyclical, and trend components using the method of Adaptive filtering.
- A program to join two different sizes of pipe at various angles to form a Y branch. The input should be only the outside diameters of the two pipes and the angle joining them. Output should be the coordinates to be plotted to make a pattern to mark the cuts on the pipes.

In order to use Newton's method on the TI-59 one must be able to write a subroutine which can evaluate the function and its derivative at any point. This task has been done in subroutine A' of figure 11 for an arbitrary order polynomial. The section of the program that performs the interative Newton's method is contained in steps 028-054. To use the program enter the order (m) of the polynomial and press A. Enter the coefficient of $x^m$ and press R/S. Enter the coefficient of $x_{m-1}$ and press R/S. Continue this process until all the coefficients have been entered (missing terms should be entered as having zero coefficients). The program contains an entry correction routine that allows the correction of a misentered coefficient. To correct an entry, enter the power of x of the misentered coefficient and press B, then enter the correct coefficient and press R/S. To complete the data entry process, enter a small positive number for $\epsilon$ and press E. Entering a first approximation of the root and pressing D will start the program. The keystrokes necessary to find the root at -10 in our sample cubic equation are shown below.

| Enter | Press |
|---|---|
| 3 | A |
| 1 | R/S |
| 5 | R/S |
| 64 +/− | R/S |
| 140 +/− | R/S |
| .00001 | E |
| 20 +/− | D |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 000 | 76 | LBL | 031 | 57 | 57 | 062 | 43 | RCL |
| 001 | 11 | A | 032 | 16 | A' | 063 | 59 | 59 |
| 002 | 42 | STO | 033 | 43 | RCL | 064 | 42 | STO |
| 003 | 59 | 59 | 034 | 56 | 56 | 065 | 58 | 58 |
| 004 | 93 | . | 035 | 94 | +/- | 066 | 73 | RC* |
| 005 | 00 | 0 | 036 | 55 | ÷ | 067 | 58 | 58 |
| 006 | 01 | 1 | 037 | 43 | RCL | 068 | 44 | SUM |
| 007 | 76 | LBL | 038 | 55 | 55 | 069 | 56 | 56 |
| 008 | 15 | E | 039 | 95 | = | 070 | 43 | RCL |
| 009 | 32 | X↕T | 040 | 44 | SUM | 071 | 57 | 57 |
| 010 | 43 | RCL | 041 | 57 | 57 | 072 | 49 | PRD |
| 011 | 59 | 59 | 042 | 33 | X² | 073 | 56 | 56 |
| 012 | 76 | LBL | 043 | 85 | + | 074 | 49 | PRD |
| 013 | 12 | B | 044 | 43 | RCL | 075 | 55 | 55 |
| 014 | 42 | STO | 045 | 56 | 56 | 076 | 43 | RCL |
| 015 | 58 | 58 | 046 | 33 | X² | 077 | 58 | 58 |
| 016 | 91 | R/S | 047 | 95 | = | 078 | 65 | × |
| 017 | 72 | ST* | 048 | 34 | ΓX | 079 | 73 | RC* |
| 018 | 58 | 58 | 049 | 77 | GE | 080 | 58 | 58 |
| 019 | 01 | 1 | 050 | 00 | 00 | 081 | 95 | = |
| 020 | 22 | INV | 051 | 32 | 32 | 082 | 44 | SUM |
| 021 | 44 | SUM | 052 | 43 | RCL | 083 | 55 | 55 |
| 022 | 58 | 58 | 053 | 57 | 57 | 084 | 97 | DSZ |
| 023 | 43 | RCL | 054 | 91 | R/S | 085 | 58 | 58 |
| 024 | 58 | 58 | 055 | 76 | LBL | 086 | 00 | 00 |
| 025 | 61 | GTO | 056 | 16 | A' | 087 | 66 | 66 |
| 026 | 00 | 00 | 057 | 00 | 0 | 088 | 43 | RCL |
| 027 | 16 | 16 | 058 | 42 | STO | 089 | 00 | 00 |
| 028 | 76 | LBL | 059 | 55 | 55 | 090 | 44 | SUM |
| 029 | 14 | D | 060 | 42 | STO | 091 | 56 | 56 |
| 030 | 42 | STO | 061 | 56 | 56 | 092 | 92 | RTN |

Figure 11
General Order Polynomial
Root Finder

**Secant Method**

The secant method is a modification of Newton's method in which the derivative has been replaced by a difference expression. This modification is helpful when the function is laborious to differentiate since the derivative does not have to be programmed. In order to allow the initial calculation of the difference expression, it is necessary to supply two different initial guesses, $x_0$ and $x_{-1}$, of the root. Using this method the expression for the new root approximation is

$$x_{N+1} - x_N = d_{N+1} = d_N f(x_N)/[f(x_N) - f(x_{N-1})].$$

The actual iterative process is shown in figure 12, and a TI-59 listing with the subroutine A' programmed to evaluate our sample cubic equation is shown in figure 13. The data register assignments for the program are listed below.

| Register | Contents |
|---|---|
| 00 | $\epsilon$ |
| 01 | $x_N$ |
| 02 | $x_{-1}$ |
| 03 | $d_{N+1}$ |
| 05 | $f(x_{N-1})$ |
| 06 | $f(x_N)$ |
| 07 and above | Available for subroutine A'. |

To use the program, enter $x_0$ and press A, enter $x_{-1}$ and press B. Enter $\epsilon$ and press E. Press D to start program execution. The keystrokes required to find the root of our sample cubic at x = 7 are shown below.

| Enter | Press |
|---|---|
| 20 | A |
| 15 | B |
| .0001 | E |
| | D |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 000 | 76 | LBL | 033 | 76 | LBL | 066 | 00 | 00 |
| 001 | 11 | A | 034 | 14 | D | 067 | 22 | INV |
| 002 | 42 | STO | 035 | 43 | RCL | 068 | 77 | GE |
| 003 | 01 | 01 | 036 | 01 | 01 | 069 | 00 | 00 |
| 004 | 91 | R/S | 037 | 16 | A' | 070 | 35 | 35 |
| 005 | 76 | LBL | 038 | 42 | STO | 071 | 43 | RCL |
| 006 | 12 | B | 039 | 06 | 06 | 072 | 01 | 01 |
| 007 | 42 | STO | 040 | 65 | × | 073 | 91 | R/S |
| 008 | 02 | 02 | 041 | 43 | RCL | 074 | 76 | LBL |
| 009 | 16 | A' | 042 | 03 | 03 | 075 | 16 | A' |
| 010 | 42 | STO | 043 | 55 | ÷ | 076 | 42 | STO |
| 011 | 05 | 05 | 044 | 53 | ( | 077 | 12 | 12 |
| 012 | 43 | RCL | 045 | 43 | RCL | 078 | 65 | × |
| 013 | 01 | 01 | 046 | 05 | 05 | 079 | 53 | ( |
| 014 | 75 | - | 047 | 75 | - | 080 | 24 | CE |
| 015 | 43 | RCL | 048 | 43 | RCL | 081 | 33 | X² |
| 016 | 02 | 02 | 049 | 06 | 06 | 082 | 85 | + |
| 017 | 95 | = | 050 | 42 | STO | 083 | 05 | 5 |
| 018 | 42 | STO | 051 | 05 | 05 | 084 | 65 | × |
| 019 | 03 | 03 | 052 | 95 | = | 085 | 43 | RCL |
| 020 | 93 | . | 053 | 42 | STO | 086 | 12 | 12 |
| 021 | 00 | 0 | 054 | 03 | 03 | 087 | 75 | - |
| 022 | 01 | 1 | 055 | 44 | SUM | 088 | 06 | 6 |
| 023 | 42 | STO | 056 | 01 | 01 | 089 | 04 | 4 |
| 024 | 00 | 00 | 057 | 33 | X² | 090 | 54 | ) |
| 025 | 43 | RCL | 058 | 85 | + | 091 | 75 | - |
| 026 | 02 | 02 | 059 | 43 | RCL | 092 | 01 | 1 |
| 027 | 91 | R/S | 060 | 06 | 06 | 093 | 04 | 4 |
| 028 | 76 | LBL | 061 | 33 | X² | 094 | 00 | 0 |
| 029 | 15 | E | 062 | 95 | = | 095 | 95 | = |
| 030 | 42 | STO | 063 | 34 | ΓX | 096 | 92 | RTN |
| 031 | 00 | 00 | 064 | 32 | X↕T | | | |
| 032 | 91 | R/S | 065 | 43 | RCL | | | |

Figure 13
Listing of Secant
Method

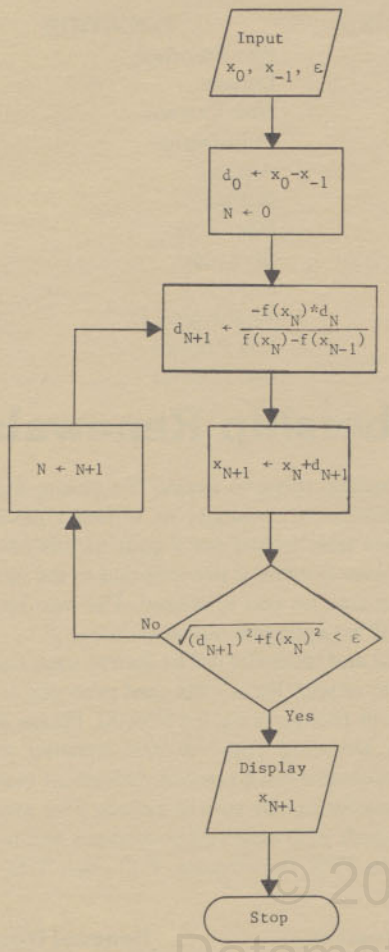


FIGURE 12
SECANT METHOD



**Conclusions**

In this article we have examined five different techniques of root finding and given examples of their application on the TI-59. No attempt has been made to single out any method as being superior to another although predictions as to the relative number of iterations required by each method have been made. The real test of a root finder, though, is not how few iterations it takes but how fast it finds the root. Since the speed of these routines is dependent on the time required to evaluate the function, which method is the fastest will depend on the function. It could turn out that even though Newton's method may take the fewest iterations to converge, this method could be the slowest of all because of the extra time required to calculate the derivative. Since speed is the major concern, one would usually use one of the root finding programs contained in the Solid State Software™ libraries because module programs execute a faster rate than their main memory counterparts. The Master Library contains a bisection program, and the Math/Utilities library contains a Newton's method program. If, however, you find occasion to need a root finder when one of these libraries is not available, the root finders presented here could be very useful.

**Related Sources**

1. **Computer-Oriented Mathematics,** Ladis D. Kovach (Holden-Day, San Francisco, 1969)

2. **Numerical Methods,** Robert W. Hornbeck (Quantum Publishers, Inc., New York, 1975)

3. **Modern Methods of Engineering Computation,** Robert L. Ketter and Sherwood P. Prowel, Jr. (McGraw-Hill, Inc., New York, 1969)

# *Précis*

*This column presents the abstracts of some of the new PPX programs which have been recently accepted. The programs were selected by our analysts as being ones that would be of special interest to our members. You can purchase these programs at a cost of $4.00 each. Send your order to: Texas Instruments: PPX Department, P.O. Box 109, Lubbock, TX 79408. Include an additional $2.00 for postage and handling plus applicable state tax.*

*If you have a need for a specific program, send a note to PPX. There is a chance that the program may have already been written. If it has, we will put the abstract in the next issue of the Exchange. Requests for programs not yet written will be placed in the "Programming Corner" column.*

**618071H Peng-Robinson Equation of State**

Calculates for a given gas via the Peng-Robinson equation of state at a specified value of pressure and temperature: specific volume, and compressibility factor, fugacity coefficient and fugacity, pressure correction to the ideal (zero-pressure) gas-phase enthalpy, and second and third virial coefficients. The required input data for the given gas includes its molecular weight, critical pressure and temperature, and acentric factor. For mixtures, the molar averages of the properties of the pure components are used.
Marcel J.P. Bogart, Whittier, CA
419 Steps

**918309H Space Attacker**

An alien fleet is printed on the tape, attacking you, the defender. Your mission is to select one of three cannons you will fire at the invaders. If you succeed a part of the fleet will disappear and the remainder of the fleet will be reprinted. Look out for the big alien leader, if you miss him, you lose.
Fred L. Hubbard, Danville, IL
478 Steps, PC-100A, Mod. 01

**148019H Loan Vs. Inflation Effect**
Given the amount of loan, interest rate, projected inflation rate, and number of payment periods, this program calculates the payment per period, sum of payments, sum of interest portions, sum of principle portions. This program calculates the effect on these three categories from the first payment to the last and sums each.
Glenn Ellis, Memphis, TN
355 Steps

**698033H A Clear-Day Insolation Model**
The user provides latitude and altitude of test site, time of year, and ground-level atmospheric dust content. The calculator will compute hourly insolation values and output them along with the time of day in local mean solar time, the solar zenith angle, and the atmospheric transmissivity. The calculator will then produce a scaled plot of insolation as a function of time.
Brad Slettene, Arcadia, CA
553 Steps, PC-100A, Mod. 10

## TI-59 Programming Seminars

There may be a seminar coming to your area. These seminars are open to anyone with a TI-59 regardless of programming background. The seminars provide both beginning and intermediate programming training on the TI-59 in a "hands on" fashion. Tuition for the two day class is $150.00 per person. This includes the instruction, workbook, and luncheon for the two days. You should supply your own TI-59. To register send your check for $150.00 payable to Texas Instruments to:

TI-59 Seminar
Texas Instruments
P.O. Box 10508 MS 5820

If you have any further questions regarding the seminars or if you would like information on setting up a company seminar, please contact Mary Ann Barley at 806—741-3272. The schedule of the upcoming seminars is listed below.

| SEMINAR DATES | LOCATION |
|---|---|
| February 18-19 | San Francisco |
| February 25-26 | Denver |
| March 4-5 | New Orleans |
| March 11-12 | Philadelphia |
| March 18-19 | Atlanta |
| March 25-26 | Buffalo |
| March 29-30 | New York |
| April 1-2 | St. Louis |
| April 8-9 | Raleigh |
| April 15-16 | Washington D.C. |
| April 20-21 | Los Angeles |
| April 22-23 | Los Angeles |

## Membership Renewals

Is your membership about to expire? To ensure that you will miss no newsletters, catalogs, or ordering privileges, check the renewal table to find out if your membership will expire soon. (If your number is not included in the range of the table, it is not time for you to renew). The next issues of the Exchange will list additional renewal dates.

A renewal card and reminder will be sent to each member before the time to renew. Return the card promptly to PPX with your check or money order for $20.00. Please do not procrastinate in returning your renewal material as our membership coordinator must remove delinquent members from our computer listing. Be sure to include your membership number on both your card and your check and mail to: Texas Instruments PPX Department, P.O. Box 109, Lubbock, TX 79408.

| Membership Number | Renewal Due: |
|---|---|
| **903932-904842** | **March 31** |
| **914842-915787** | **March 31** |
| **924326-924863** | **March 31** |
| **930282-930488** | **March 31** |
| **904843-906124** | **April 30** |
| **915788-916789** | **April 30** |
| **924864-925575** | **April 30** |
| **930489-930683** | **April 30** |



TEXAS INSTRUMENTS
INCORPORATED
PPX • P.O. Box 53 • Lubbock, Texas 79408
U.S. CALCULATOR PRODUCTS DIVISION

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. POSTAGE
PAID
Permit No. 1
Lubbock, Texas