NEWTON – Finding Roots of Equations

Abstract

NEWTON is a simple program written in 1980 for the SHARP PC-1211 pocket computers and compatibles to find roots of an arbitrary user-supplied equation f(x)=0 using Newton's method and a user-provided initial guess. Two worked examples are included.

Keywords: *root finding, solving equations, Newton's method, SHARP pocket computers, PC-1210, PC-1211, PC-1212, compatibles*

1. Introduction

NEWTON is a very simple 3-line *(~150-byte)* practice program that I wrote in 1980 for the *SHARP PC-1211* pocket computer and compatible models (runs *as-is* in the *PC-1210* and *PC-1212* and with minimal or no changes in many other models), which will try to find a real root of an user-supplied equation *f(x)=0* using *Newton's method* and a user-provided initial guess.

The procedure is as follows: given an equation *f(x)=0* and an initial guess for the root, x_0, *Newton's method* produces a hopefully improved guess, x_1, computed this way:

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

where *f'(x)* is the derivative of *f(x)*, which is numerically approximated like this:

$$f'(x) \sim \frac{f(x+h) - f(x)}{h}$$

where *h* is a small value, for instance *h=0.00001* usually gives good results for a 10-digit machine. The process is iterated with x_1 replacing x_0 to produce a further improved guess x_2 and so on until it either *converges* to the root *x*, *diverges* to infinity or *enters* a cycle. In these last two cases the user must stop the execution *(by pressing the* **BRK** *key)* and either re-run the program with a different initial guess x_0 or decide that no real root exists.

2. Program Listing

```
10: "A" INPUT "X INIT=";T: H=0.00001
20: X=T: GOSUB 40: Z=Y,X=T+H: GOSUB 40: S=T,T=S-HZ/(Y-Z+(Y=Z)): IF T<>S PAUSE T: GOTO 20
30: BEEP 1: X=T,Y=Z: PRINT "ROOT=";X: PRINT "F(X)=";Y
```

3. Usage Instructions

See the worked examples to understand how to use the program.

4. Examples

The following examples can be useful to check that the program is correctly entered and to understand its usage.

4.1 Example 1

Find a root of *Leonardo de Pisa*'s cubic equation: $x^3 + 2x^2 + 10x = 20$

In **PRO** Mode, enter the following program line to define the function *f(x)* to be solved:

```
40: Y=XXX+2XX+10X-20: RETURN
```

In **DEF** Mode, proceed as follows to find a root, using ***1*** as initial guess:

SHFT **A**	→	X INIT=_
1 **ENTER**	→	1.411764706 → 1.369336593 → 1.36880819 → 1.368808108 *(all briefly paused)*
		ROOT= ***1.368808108***
ENTER	→	**F(X)=** *0.*

Each iteration is shown, the root found is correct to all 10 digits and the resulting *f(x)* indeed evaluates to **0**.

4.2 Example 2

Find *two* roots, large and small, of the 5th degree equation: $16x^5 - 180x^3 + 405x - 136 = 0$

In **PRO** Mode, enter the following program line to define the function *f(x)* to be solved:

```
40: Y=16*X^5-180*X^3+405*X-136: RETURN
```

In **DEF** Mode, proceed as follows to find 2 roots, large and small, using ***4*** and ***0*** as initial guesses, respectively:

SHFT **A**	→	X INIT=_	
4 **ENTER**	→	3.481588554 → 3.14982053 → ... → 2.942932637	*(all briefly paused)*
	→	**ROOT=** ***2.942932637***	*(the "large" root)*
ENTER	→	**F(X)=** *0.00000083*	*(non-zero but adequately small for such large root and coefficients)*
SHFT **A**	→	X INIT=_	
0 **ENTER**	→	3.358024691E-01 → 3.553537317E-01 → ... → 3.555553918E-01	*(all briefly paused)*
	→	**ROOT=** ***3.555553918E-01***	*(the "small" root, x = 0.3555553918)*
ENTER	→	**F(X)=** *0.*	

Notes

1. If a real root exists *Newton's method* usually converges *quadratically* to it, i.e. once the convergence starts the number of correct digits *doubles* after each iteration, unless the root's multiplicity is >1 in which case the convergence reduces to *linear*.

2. After each iteration the value of the current guess is briefly paused, which is useful to determine if convergence has started or else the values *diverge* to infinity or are stuck in a *cycle* (periodic or not) so the user can then decide to abort the execution.

3. Once a root is found both the *root* and the value of *f(x)* at the root (which must be ***0*** or nearly so) are displayed and execution stops. Also, they are stored in variables *X* and *Y*, respectively, so that they can be reused in further calculations.

4. If evaluating the derivative *f'(x)* ever results in ***0*** a *division by 0 error* would ensue, but the program avoids it by using the value ***1*** instead so that no error arises and the search moves on to another place. This usually happens at a *minimum* of *f(x).*

References

Francis Scheid (1988). *Schaum's Outline of Theory and Problems of Numerical Analysis, 2nd Edition*.

Copyrights