Finding Roots of Polynomials

Tom James

May 4, 2012
Contents

I  Introduction  2

1  Prerequisite Skills  2
   1.1  Math material prerequisites  2
   1.2  Calculator programming prerequisites  2

II  Student Version  3

2  Introduction  3

3  Bisection Method  3
   3.1  Exercises  3
   3.2  Assignment  4

4  Test your program  4

III  Teacher’s Version  5

5  Bisection method algorithm  5

6  Sample solution  5
   6.1  Error-proofing  6
   6.2  Sample execution  6

7  Anticipated trouble spots  6

8  Extensions  7
   8.1  Secant method  7
   8.2  Precision  7
   8.3  Running time  7
Part I

Introduction

This module will facilitate student understanding of several root-finding algorithms for functions by guiding students through the process of implementing those algorithms on their graphing calculators using the programming language TI-BASIC. Students will begin by investigating polynomial roots and then apply their algorithm to general continuous functions.

1 Prerequisite Skills

This module would be appropriate for students in a precalculus or second-year algebra class.

1.1 Math material prerequisites

Polynomials Students are familiar with polynomials of arbitrary degree. Students recognize polynomials as functions of an independent variable and are able to use the Factor Theorem to relate the root of a polynomial to a linear factor of that polynomial. Furthermore, students are comfortable interpreting and using the following vocabulary words:
- Root, zero, x-intercept
- Degree
- Continuous

Intermediate Value Theorem Students are able to use the result of the Intermediate Value Theorem to justify the existence of a root \( c \) in between \( a \) and \( b \) if \( f(a) \) and \( f(b) \) have opposite signs. However, students may not know they are referencing this particular theorem by name.

This is a more advanced program for students to write and should only be attempted with students who have written programs on their calculator before.

1.2 Calculator programming prerequisites

Basic syntax Students recognize the need to close any structure that is opened, whether it is an open quote, open parenthesis, or beginning of a conditional statement.

Use of variables Students are able to assign, reference, and copy the value of a variable. They are able to create a counter variable that is incremented by 1 after each pass through a loop.

Use of basic logical expressions Students are able to use the logical operators “and” and “or.”

Conditional statements Students are able to construct an “If – Else If – Else” conditional statement

Loops Students are able to construct a “while” loop given a condition under which the loop should continue.
Part II

Student Version

2 Introduction

This module is designed for you to explore a simple question with a striking variety of solutions: how do we find a root $r$ of a given function $f(x)$?

For many functions, we can factor and find the roots explicitly. However, there are many more functions that cannot be easily broken down into linear or quadratic factors. For these functions, we have several tools at our disposal for locating the zeros. These take the form of procedures for starting somewhere on the curve and steadily getting closer to a root using a defined algorithm.

3 Bisection Method

Our basic idea when looking for a root using this method is to keep subdividing the $x$-axis until we have either found our root or gotten sufficiently close to it.

Suppose $f(x)$ is a continuous function and there exist $a$ and $b$ such that $f(a)$ and $f(b)$ have opposite signs. So we’ll be looking for a point $c$ in between $a$ and $b$ where $f(c) = 0$, which we know must exist by the Intermediate Value Theorem.

Here is the basic algorithm:

1. Suppose $l$ (our initial left boundary) and $r$ (our initial right boundary) are two points on the $x$-axis with $l < r$, where $f(l) < 0$ and $f(r) > 0$.
2. Let $m = \frac{l + r}{2}$.
3. If $f(m) = 0$, then $m$ is a root and we’re done.
4. We haven’t yet found a root. If $f(m)$ has the same sign as $f(l)$, then set $l = m$.
5. Otherwise, set $r = m$.
6. Go back to step 2.

Notice the distance between $l$ and $r$ is cut in half each time through the loop. In other words, the size of our “window” around the root will shrink to $\frac{r - l}{2}$ at the end of each pass.

3.1 Exercises

Exercise. Run the bisection method algorithm by hand for $f(x) = x^3 - 9x$. Let $l = 1$ be your initial left boundary and $r = 9$ be your initial right boundary. Write out your successive values of $l$, $r$, and $m$.

- Which root did you find?
• What initial boundaries might you pick to find the two other roots?

**Exercise.** Run the bisection method algorithm by hand for \( f(x) = x - 1 \). Let \( l = -2 \) be your initial left boundary and \( r = 3 \) be your initial right boundary.

• What do you notice about your successive guesses for \( m \)?

• Will \( m = 1 \) ever?

• Looking at the sequence of values of \( m \), at what point would you be satisfied that \( x = 1 \) is a root of \( f(x) \)?

### 3.2 Assignment

Your first program should implement this algorithm. It should take as input a function \( f(x) \), initial left and right boundaries \( l \) and \( r \), and an acceptable error amount (or tolerance) \( T \). It should output the value of the root \( x \) as well as the value of \( f(x) \).

Here is a list of commands you will find useful:

**Logical Operator OR** If you want to test if at least one of two conditions is true, use \( \text{2nd MATH LOGIC 2: or} \).

### 4 Test your program

**Exercise.** Approximate the value of \( \sqrt{2} \) to within 0.0001 by using your program to find the positive root of \( f(x) = x^2 - 2 \). Then, count how many passes through the loop were necessary by adding a counter variable that gets incremented by 1 at the end of each pass through the loop.

**Exercise.** Will the bisection method always yield a root of \( f(x) \)? Why or why not?

**Extension.** What can you say about the maximum distance your guess could be from the root after \( n \) passes through the loop?
Part III

Teacher Version

5 Bisection method algorithm

Because of the limitations introduced by this method, we cannot always expect to precisely locate a root of a given polynomial. However, we can get within a tolerance $T$ of it.

Here is the basic algorithm students will follow:

1. Suppose $l, r \in \mathbb{R}$ with $f(l) < 0$ and $f(r) > 0$ and $T > 0$ is our desired maximum distance from the root.
2. Let $m = \frac{l + r}{2}$.
3. If $f(m) = 0$, then $m$ is a root and we’re done.
4. If $\frac{r - l}{2} < T$, then we’re within our error tolerance; $m$ is a root and we’re done.
5. We haven’t yet found a root. If $f(m)$ has the same sign as $f(l)$, then set $l = m$.
6. Otherwise, set $r = m$.
7. Go back to step 2.

6 Sample solution

Precondition: $f(l) < 0, f(r) > 0$

This solution uses an infinite while loop to continue bisecting the $x$-axis until a root is found in an $\epsilon$-neighborhood (represented by the variable $T$) of $x$.

Note that $Y_1$ must be surrounded by quotes when entered on the calculator.

```plaintext
1 Prompt L
2 Prompt R
3 Prompt Y₁
4 Prompt T
5 While 1
6 (L + R) / 2 → M
7 M → X
8 Y₁ → F
9 L → X
10 Y₁ → G
11 If F=0 or (R - L) / 2 < T
12 Then
13 M → X
14 Disp X
15 Disp ‘ROOT FOUND’
16 Disp Y₁
17 Stop
```
18 Else : If (F>0 and G>0) or (F<0 and G<0)
19 Then
20 M → L
21 Else
22 M → R
23 End
24 End
25 End

6.1 Error-proofing

Because of the precondition that \( f(l) \) and \( f(r) \) have opposite sign, students may want to build in a quick check near the beginning of their program to avoid bad results based on invalid input. For example, this code snippet, inserted near the beginning of the program, checks to see if the product \( f(l) \cdot f(r) \) is positive, which occurs if and only if the two factors have the same sign:

1...  
2  L → X  
3  Y1 → U  
4  R → X  
5  Y1 → V  
6  If U*V ≥ 0  
7  Then  
8  Disp 'ERR:ENDPOINTS'  
9  End  
10...

6.2 Sample execution

1  L=? 0  
2  R=? 2  
3  Y1=? 'X^2-2'  
4  T=? 10^-5  
5  1.414207458  
6  ROOT FOUND  
7  -1.72643337E-5  
8  Done

7 Anticipated trouble spots

- Students may need a reminder on how to insert a blank line of code in the middle of their program using \[\text{2nd} \] [DEL] [ENTER].
- Students need to remember to surround their function in quotes when inputting into the program.
- When multiple roots are close together within a given interval, students need to take care to set their boundaries to distinguish them.
8 Extensions

8.1 Secant method

If students are able to implement and master the bisection method, they may wish to explore the secant method. This method can be easily implemented on the calculator by modifying the bisection method program.

For advanced students who are able to use both techniques, Dekker’s method provides a natural way to combine the bisection method and secant method in order to improve the order of convergence of the sequence.

8.2 Precision

Given the limitations of the floating-point number representation system implemented on handheld computing devices, students may be able to construct cases in which the bisection algorithm fails to identify a root. A natural extension would be to explore these cases, why they break down, and alternate methods for locating a root requiring many digits of precision.

8.3 Running time

Students can use counter variables to easily determine how many passes through an algorithm are required to arrive at an answer within a certain tolerance. Students can compare the running time of the bisection method, secant method, and Dekker’s hybrid method as a precursor to big-O notation.