Lecture 11: Basic Maple Programming

1. The numerical solution of equations

Let \( f : [a, b] \rightarrow \mathbb{R} \) be a given function. We consider the problem of finding a solution \( x \in [a, b] \) of the equation

\[
\text{(E)} \quad f(x) = 0.
\]

We shall assume that \( f \) is continuous and that

\[
\text{(C)} \quad f(a)f(b) < 0.
\]

This last condition implies that \( f \) takes both positive and negative values in \([a, b] \). By continuity, \( f \) must also vanish somewhere in the interval (cf. the Intermediate Value Theorem in Analysis). Hence Equation (E) has a solution.

The bisection method for computing a solution defines two sequences \( \{a_n\}_N \) and \( \{b_n\}_N \) by recurrence as follows: start with

\[
a_0 = a \quad \text{and} \quad b_0 = b.
\]

Then, by our earlier assumption,

\[
f(a_0)f(b_0) < 0.
\]

Now let \( n \in \mathbb{N} \) and suppose that

\[
f(a_n)f(b_n) < 0.
\]

This means that \( f(a_n) \) and \( f(b_n) \) have opposite signs. Define

\[
\overline{x} = \frac{a_n + b_n}{2}.
\]

Then there are three possibilities: either

\[
(1.1) \quad f(\overline{x}) = 0
\]

or

\[
(1.2) \quad f(a_n)f(\overline{x}) > 0
\]

or

\[
(1.3) \quad f(a_n)f(\overline{x}) < 0.
\]

In the first case, \( \overline{x} \) is a solution of Equation (E) and the calculation terminates.

In the second case, \( f(\overline{x}) \) has the same sign as \( f(a_n) \), and so we set

\[
a_{n+1} := \overline{x} \quad \text{and} \quad b_{n+1} := b_n.
\]

In the third and final case, \( f(\overline{x}) \) and \( f(a_n) \) have opposite signs, and so we set

\[
a_{n+1} := a_n \quad \text{and} \quad b_{n+1} := \overline{x}.
\]

With \( a_{n+1} \) and \( b_{n+1} \) defined in this way, a moment’s reflection shows that

\[
f(a_{n+1})f(b_{n+1}) < 0.
\]
Hence there is a solution of Equation (E) between \(a_{n+1}\) and \(b_{n+1}\) and

\[
|a_{n+1} - b_{n+1}| = \frac{1}{2} |a_n - b_n|.
\]

In other words, the length of the interval where we know that there is a solution has been reduced by half. Unless the calculation terminates because we have found a solution, we obtain at the \(n\)th iteration, that a solution exists between \(a_n\) and \(b_n\), where

\[
|a_n - b_n| = 2^{-n}(b - a).
\]

One could say that the bisection method finds an additional digit in the binary expansion of the solution with each new iteration.

**Example 1.1.** We use the bisection method to compute \(\sqrt{2}\). Let \(f: [1, 2] \rightarrow \mathbb{R}\) be defined by

\[
f(x) = x^2 - 2.
\]

The first few iterates, in decimal form, are shown in Table 1.

<table>
<thead>
<tr>
<th>(n)</th>
<th>(a_n)</th>
<th>(b_n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>1.375</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>1.375</td>
<td>1.4375</td>
</tr>
<tr>
<td>5</td>
<td>1.40625</td>
<td>1.4375</td>
</tr>
</tbody>
</table>

Table 1. The \(a_n\) and the \(b_n\) found by the bisection method for Example 1.1.

2. **MAPLE Functions and Procedures**

The bisection method is conceptually simple, yet its implementation gives rise to a number of practical issues:

- **if-then-else** statements are necessary.
- A loop is required. How many iterations?
- Exact or floating-point arithmetic?
- How do we arrange so that the implementation works for any (mathematical) function that satisfies Condition (C)?

**MAPLE** provides various means of implementing the mathematical concept of function. For elementary mathematical functions such as

\[
f(x) = x^2 - 2
\]

the “arrow construction” is simplest:

\[
f := x \rightarrow x^2 - 2:
\]

Then the value of the function at, say, 1 can be obtained by issuing the command

\[
f(1);
\]

The bisection method itself can be implemented as a procedure with the following arguments:

- \(a\), the left endpoint of the initial interval.
- \(b\), the right endpoint of the initial interval.
- \(f\), the function in Equation (E).
- \(N\), the number of iterations to perform.

The procedure modifies \(a\) and \(b\) so that when they are returned \(a = a_N\) and \(b = b_N\).
3. The bisection procedure

\[
\text{bisection} := \text{proc}(a,b,f,N) \ni \text{local } n,x,p,A,B : \\
\text{if } (\text{evalf}(f(a))=0) \text{ then} \\
\quad \text{return } a : \\
\text{elif } (\text{evalf}(f(b))=0) \text{ then} \\
\quad \text{return } b : \\
\text{else} \\
\quad \text{fi} : \\
\text{p} := \text{evalf}(f(a)*f(b)) : \\
\text{if } (p > 0) \text{ then} \\
\quad \text{print ('The sign is the same at both points!') :} \\
\quad \text{return } a,b : \\
\text{else} \\
\quad \text{fi} : \\
\text{A} := a : \\
\text{B} := b : \\
\text{for } n \text{ from 1 to } N \text{ do} \\
\quad x := \text{evalf}(0.5*(A+B)) : \\
\quad p := \text{evalf}(f(x)*f(A)) : \\
\quad \text{if } (p=0) \text{ then} \\
\quad \quad \text{return } x : \\
\quad \text{elif } (p > 0) \text{ then} \\
\quad \quad A := x : \\
\quad \text{else} \\
\quad \quad B := x : \\
\quad \text{fi} : \\
\text{od} : \\
\text{return } A,B : \\
\text{end proc} :
\]