Mathematics 371 – Numerical Analysis
Lab 1/Problem Set 2 – First Steps in Maple Programming for Numerical Computing
September 10, 2001

Background

Note: All of the Maple data structures and commands used here are also illustrated in the Maple worksheet `Primer.mws` *which can be viewed from the course homepage. That worksheet also contains some examples computed with the Bisect procedure and you may want to refer to it as you are working through the following examples and exercises.*

Maple has a large collection of built-in commands for performing various mathematical computations, including a number of routines for numerical root-finding, integration, solution of algebraic and differential equations, etc. In addition, when we want to implement a numerical algorithm ourselves (for instance, when that algorithm is not built-in), we can use the Maple *programming language* to code and implement our own new commands or procedures extending the basic functionality of the program. This is an imperative or procedural language similar in spirit and syntax to Pascal and, to a lesser degree, C. It does not incorporate objects, etc. as in C++. In this lab meeting, we will work through the development of a Maple procedure for the bisection root-finding method we have been discussing.

Getting the Maple Code for the Bisection Method

First, carry out the following steps. In each case, the command will have been successfully carried out if you see no output/error messages, etc.

1. Open a Text Editor window (click right mouse button with cursore over the workspace background, and select Text Editor with left mouse button).
2. Launch a Terminal window also, and enter the command: `netscape &` after the prompt (and ENTER). (The ampersand in the command means to launch netscape in a separate window and return to a new command prompt in this window. Without it, this window would be "busy" as long as the netscape window was open, and you could not enter further commands from it.)
3. Go to the course homepage:

   `http://mathcs.holycross.edu/~little/Num01/Num01.html`

   and follow the link *Maple code for bisection method.*
4. Highlight and Copy/Paste the whole procedure from the browser window into the Text Editor window (use the left keypad buttons on the keyboard for Copy and Paste).
5. Save the contents of the Text Editor window to a file in your directory (File/Save). You can call the file anything you want. I usually use the extension .map in files containing Maple source code, but that's not a required way to name these files.
6. From the Terminal window, launch Maple 7 with the command `xmaple &`.
7. In the Maple window, enter a command like this:

```
read "filename";
```

where "filename" is the full name you gave the file when you saved it in step 5.

You should now have a working copy of the Bisection method procedure, read into Maple so that it can be used. Here is some information about the procedure, how it is used, and how it works.

1. The first line

   ```
   Bisect := proc(f,a,b,tol)
   ```

   is called the *procedure heading*. It specifies the way the procedure will be "called" or invoked as a command to perform a calculation. The items `f,a,b,tol` are called the parameters for the procedure – they are the information that must be provided in order for the procedure to do its job – the function $f(x)$, the endpoints of the original interval, and the tolerance (which determines when the bisection process will terminate).
2. Once the procedure source code is read in with the `read` command, this procedure can be called by defining a function, then issuing the Bisect command, like this (enter this in your Maple session). For example, suppose we wanted to determine the root of the equation $x - \tan(x) = 0$ in the interval $(\pi, 3\pi/2)$. First we define the function:

   ```
   f := x -> x - tan(x);
   ```

   The function $\tan(x)$ has a vertical asymptote at $3\pi/2$, so we will stop just a bit to the left of there, say at $299\pi/200$. Then, to compute a value of the root correct to 6 decimal places, the Bisect procedure can be called with this statement:

   ```
   Bisect(f,evalf(Pi),evalf(299*Pi/200),.0000005);
   ```

   Note: As we saw before, the `evalf` function gives the floating point form of the number supplied to it. *The endpoints of the interval should be supplied to the procedure in this form.* We are using Maple's default 10-digit rounding floating point numbers. The output is the approximate solution: $x \doteq 4.493409661$, which is correct to 6 decimal places.
3. The next lines starting with # characters are *comments* – they document how the procedure works, but are not part of the code executed by Maple.
4. The line `local left,right,p,fl,fr,fp;` defines local variables used while the procedure is running.
5. The next four statements are *assignments*. We take the value of `a` and assign that to a variable `left` (the left endpoint of the current interval being searched), store the value of $f(a)$ in `fl`, and similarly for the right endpoint.
6. The next `if` statement tests whether the function changes sign on the interval $[a, b]$. If not, the procedure does not try to find a root (since there may not be one), and prints an error message. For instance, try:

   ```
   Bisect(f,evalf(Pi),evalf(5*Pi/4),.0000005);
   ```

7. The `while` statement in the `else` branch is a *loop*, or repetition structure. The "body" of the loop – that is, the statements up to the `end do;` – will be repeated as long as the condition `(right - left)/2 > tol` is true. The loop will terminate the first time that condition becomes false. (That will happen eventually, since we are cutting down `right - left` by a factor of 2 in each pass through the loop.)
8. The "body" of the loop consists of the steps we discussed in class – we compute the midpoint, and the value of f at the midpoint. If that value is "effectively zero" ($|f(p)| < .1 \times 10^{-19}$ – an arbitrary choice), then we return the value of p as the approximate root, otherwise we find one half of the current interval on which f changes sign and update the left or right endpoints accordingly.
9. The `return` statements specify the output of the procedure – either the value of p itself if $|f(p)| < .1 \times 10^{-19}$ at any step, or the midpoint of the current interval if the loop runs until `(right - left)/2` is less than or equal to `tol`. When a `return` statement is executed, the procedure terminates, and returns you to the Maple input prompt.

Lab 1/Problem Set 2

A) Using the Bisect procedure, do problem 7 from Section 2.1 in the text.
B) Problem 9 from Section 2.1 (try to understand how this will work by "pure thought" first, then use the Bisect procedure to check yourself).
C) Use the procedure to do problems 11, 13 from Section 2.1.
D) Problem 16 from Section 2.1. (This is also "pure thought" problem.)
E) An alternate way to code the Bisection method would be to use our discussion showing how to predict the number of bisection steps needed to get an approximation good to a certain accuracy (tolerance). Develop a modified version of `Bisect` that first computes the number of steps needed, then uses a *count-controlled* (`for`) loop rather than a `while` loop to do that number of bisections. (See the worksheet *Primer.mws* or the Maple online Help for more info about count controlled loops.) Suggestions: There is a built-in Maple function called `ceil` (for "ceiling") that takes a floating point number and returns the next biggest integer. That may be useful here! Also, try to reuse as much of the given Bisect procedure as possible. Copy and paste a second copy of that procedure in your source code file, and modify the copy to create the procedure asked for here. Name it something other than *Bisect* so that you have two different procedures to compare. Test your procedure on the equations from problem 7.
F) Do the following problems on fixed point iteration by hand: Section 2.2/1-3,7,11ac,13,18.

Assignment

Individual solutions due in class Monday, September 17.