Spring 1998

# Maple Lab, Week 18

## Newton's Method and Maple Programming

**Background:** The objective of this lab is to use Maple to illustrate Newton's method, an interative method for finding approximate solutions to the equation $f(x) = 0$ :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 1, 2, \ldots.$$

In addition, this will be an excellent opportunity to introduce some commands useful for programming with Maple, such as `for .. do .. od;` for writing loops of commands (REMARK: `od` is `do` spelled backwards, and ends the loop), the `if .. then .. fi;` command (REMARK: `fi` is `if` spelled backwards, and ends the `if` command), and the `proc() .. end;` command for writing procedures. The exercises below will be based on Exercise 11 in Chapter 3 of your CalcLabs Manual, solving $f(x) = x^3 + x - 2 = 0$.
In preparation for the exercises, execute the following Maple commands:

```
> f := x -> x**3 + x - 2; fp := D(f);
> Newton := x -> evalf(x - f(x)/fp(x));
> Digits := 20:  tol := 10.**(-Digits + 2);
```

Make a plot of `f` to locate all the roots. (It is easy to see that, in this case, the only root is $x = 1$. This will make it easy to see how accurate the approximations are.)

**Exercises: 1.** Execute the command `Newton(0);`, and then `Newton(");` a few times, until the output converges to 1. Retype the command each time so that all iterations are left on the screen. What is the number of digits of accuracy of each approximation?

**2.** The `for .. do .. od;` command.
The above sequence of commands can easily be executed in a loop as follows:

```
> x := 0:
> for i from 1 to 8 do
> x := Newton(x);
> od;
```

Now repeat the above commands, but replace `Newton(x)` by `x - f(x)/fp(x)`. Explain the result.

**3.** The `if .. then .. fi;` command.
Execute the following version of the first loop in Exercise 2, in order to halt the computation when successive iterates are sufficiently close, i.e., when $|x_{n+1} - x_x|$ is sufficiently small: Replace the line `x := Newton(x);` by the following three commands: `x1 := Newton(x); if abs(x1-x)<.0001 then break fi; x := x1;`. Explain the result.

**4.** The `proc() .. end;` command.
A Maple **procedure** is a group of Maple commands which form what's usually referred to in programming languages as a **subprogram**. It would be called a **subroutine** in FORTRAN, or a **function** in C. It typically has input variables, one or more outputs, and possibly some internal variables that are known as **local** variables. As an example, the following is a procedure to calculate the circumference and area of a circle of radius `r`.

```
> Circle := proc(r)
> local C,A;
> C := evalf(2*Pi*r);
> A := evalf(Pi*r**2);
> RETURN(C,A);
> end;
```

To use the procedure, say to find the circumference and radius of a circle of radius 10, the user enters

```
> Circle(10);
```

Your lab task is to write a procedure named `Newt` that implements Newton's method. Have as input the function `Newton`, as defined above, the starting value `xstart`, and the tolerance, `tol`. Use a `for .. do .. od;` loop as in Exercise 4, with a maximum of 10 iterations, and stop when successive iterates are within `tol`. Return the value `x1` of the last iteration. Test your procedure with the command

```
> Newt(Newton,0,tol);
```

**Do either Exercise 5 or Exercise 6; extra credit for doing both.**

**5.** Apply the software you have developed, to find all the real roots of $10x^4 + 4x^3 - 0.4x^2 - 0.01x + 0.007 = 0$. Comment on the importance of a good starting value.

**6.** You might be asking, why bother learning about Newton's method, when Maple has a command for solving equations, `fsolve`? Of course, the easy answer is that you may not always have Maple available to you. Another answer is that Newton's method might be faster than `fsolve`, and if so, this could be significant if you have a lot of equations to solve. This exercise will be to use Maple's `time` command to find out which is faster, `fsolve` or `Newt`, in solving the equation $f(x) = 0$.
Use the on-line help for the `time` command. Next, find out how long it takes to execute the command `Newt(Newton,0,tol):` 100 times. (Remark: You should use a loop, and end all statements with a colon, to suppress the output). Then, find out how long it takes to execute the command `fsolve(f(x)=0,x):` 100 times. NOTE: You might have to unassign `x` before using `fsolve`.