CURTIN UNIVERSITY
DEPARTMENT OF MATHEMATICS & STATISTICS

ACCELERATED MATHEMATICS UNITS MATH1017 AND MATH1021

Lab Sheet 7

This week we want you to code a programming function (`proc`) that will do Newton's method for finding approximate roots of an equation. There's a mini i-lecture on it, but it's probably a bit late for you to look at it now.

Since the word "function" is used by programmers and mathematicians and mean similar but different things, and both usages occur below, we will say *programming function* when *function* is used in the computing/programming sense and *maths function* when *function* is used in the mathematical sense. Indeed, we have already used *programming function* above.

So what we want you to do is to code up the algorithm, test it, and then login to `AM M1017 (Super)` or `AMe M1021 (Super)` at the usual place

`http://aim03.curtin.edu.au`

Your password is the first 4 characters of your usual AiM password together with a 4 character "word" from your tutor. So, if you usual password starts with `fred` and the "word" your tutor gives you is `boat`, then your super password would be

`fredboat`

Once you've logged in do the question there, and yes you can use the function you've coded to enter your answers.

Background

Essentially, given a maths function f and an inital estimate x_1 for a root of

$$f(x) = 0,$$

and assuming that the Newton-Raphson method has been applied $n-1$ times, then the next estimate x_{n+1} is given by

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

So, in particular,

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

Ingredients: `proc`, using a `repeat` loop.

If you recall the first lab, Maple doesn't provide a `repeat` loop, but it can be emulated doing a `break` from an infinite loop.

Also, we told you how Maple does "one-line" functions just like maths functions using *map* syntax. So if f is the *square* function, i.e. $x \mapsto x^2$, then you can write in Maple,

```
f := x -> x^2
```

Of course the derivative function of f is defined by $f'(x) = 2x$, which as a *map* is $x \mapsto 2x$. Wouldn't it be nice if there was a function in Maple would take a function written as a map, and return its derivative also as a map. There is such a function, it's `D`. Verify that

```
D(f)
```

does indeed return the derivative of a function f defined as a map, as a map.

Here is a suggested approach.

1. Call your function `newton` (or something you like). It will take arguments: function `f` (and we will expect it do be defined using *map* syntax), an initial guess `x1`, and a tolerance `toler` (which will not really be how close we are to the actual root, but instead is the magnitude of $|f(x_n)|$ we will tolerate and so accept x_n as close enough to the actual root).
   Note. The heading line won't line up perfectly, but we really don't have time to fuss with it. It's good enough.

```
newton := proc(f, x1, toler)
            local Df, xi, fxi, Dfxi, nextxi, i;
            xi := x1;
            Df := D(f);
            printf("  i   |    xi   |  f(xi)  |  f'(xi)   |  xi+1  \n");
            printf("------------------------------------------------\n");
            for i do
               fxi  := f(xi);
               Dfxi := Df(xi);
               nextxi := xi - fxi/Dfxi;
               printf(" %4d | %6e | %6e | %6e | %6e\n",
                      i, xi, fxi, Dfxi, nextxi);
               xi := nextxi;
               if is(abs(f(xi)) < toler) then
                 break;
               fi;
            od;
            return evalf(xi);
          end;
```

2. Try it out with $f(x) = x^3 - 2x - 5$ and $x_1 = 1$, i.e. you will need to call it this way.

```
f := x -> x^3 - 2*x - 5;
newton(f, 1, 1e-4);
```

   where a tolerance of 1×10^{-4} has been assumed.

3. Now login and do the quiz!!