C PROGRAMMING: ROOT FINDING BY BISECTION

We have a few specialized equations like the quadratic formula to find the values x for which $f(x) = a\,x^2 + b\,x + c = 0$. How would we solve an equation like $\tan(x) - 2x = 0$? (This is called a transcendental equation.) One answer is the bisection method.

Suppose you know that $f(R) < 0$ and $f(L) > 0$. Then, assuming f has no breaks in it, there must be a root x between R and L. To find x, we compute the value of f at the midpoint $M = (R + L)/2$. Then we see if $f(M)$ and $f(R)$ have the same sign. If they do, then we can choose M as the new R, since we know the root is not between R and M. On the other hand, if $f(M)$ and $f(R)$ have opposite sign, we choose M as the new L.

The bisection method is implemented for a quadratic function in the code on the next page. We start with this case, where we already have the quadratic formula, so we can check it works.

The bisection method is a very good method for finding roots, but it does require that you know two values R, L between which f changes sign. Also, if there are several roots between R and L you will only find one of them.

We are writing this code for several reasons. First, because it is another illustration of the utility of do-while loops. In the bisection method we don't really know how many steps it will take to get the root. So we keep going until L and R are less than some very small number ϵ apart.

Second, this code illustrates the use of a function. You will notice that in addition to main it has a second piece double myFunction($\cdots$) { $\cdots$ } The first set of $\cdots$ are any numbers you want to send this new piece of code from main. The second set of $\cdots$ are the instructions which evaluates the function which is returned by it to main.

Why do we use functions? The reason is that we may want to evaluate the same mathematical expression at many locations in a program. If so, it is easier and safer to type it out once and for all in some separate part of the code. This is especially true if you anticipate changing the function some time later. You only have to change it in one place.

Get the program running and check it against the quadratic formula. (Eg if you use $a = 3, b = 8, c = -35$ and starting left point -1 and starting right point 5 you will get the root

$x = 2.6666667$). Then change the function to $\tan(x) - 2x = 0$. (You no longer need to read in a, b, c or pass them to the function. See page 3 for the new version.) Try starting with left point 0.2 and right point 1.5 and see what you get.

```
/* This program computes root of function with bisector method  */
#include <stdio.h>
#include <math.h>
double myFunction(double x, double a, double b,double c){
   return a*x*x + b*x +c;
}
int main(void){
   double a,b,c;
   double leftpt, rightpt, midpt, epsilon = .0000001;
   double midvalue, rtvalue, root;
   printf("\nEnter values for a,b,c:\n");
   scanf("%lf %lf %lf", &a, &b, &c);
   printf("\nEnter values for starting left and right points:\n");
   scanf("%lf %lf", &leftpt, &rightpt);
   printf(" Left and right starting points are: %lf , %lf\n", leftpt,rightpt);
   do {
      midpt = (leftpt + rightpt)/2;
      rtvalue = myFunction(rightpt,a,b,c);
      midvalue = myFunction(midpt,a,b,c);
      if (rtvalue * midvalue >= 0)
             rightpt = midpt;
      else   leftpt  = midpt;
   } while ((rightpt - leftpt) > epsilon);
   root = (rightpt+leftpt)/2;
   printf("\nRoot for equation %5.2lf *x**2 + %5.2lf *x + %5.2lf  is:",a,b,c);
   printf("  %15.10lf\n", root);
   return 0;
}
```

```
/* This program computes root of function with bisector method  */
#include <stdio.h>
#include <math.h>
double myFunction(double x){
   return tan(x) -2.*x;
}
int main(void){
   double leftpt, rightpt, midpt, epsilon = .0000001;
   double midvalue, rtvalue, root;
   printf("\nEnter values for starting left and right points:\n");
   scanf("%lf %lf", &leftpt, &rightpt);
   printf(" Left and right starting points are: %lf , %lf\n", leftpt,rightpt);
   do {
      midpt = (leftpt + rightpt)/2;
      rtvalue = myFunction(rightpt);
      midvalue = myFunction(midpt);
      if (rtvalue * midvalue >= 0)
             rightpt = midpt;
      else   leftpt  = midpt;
   } while ((rightpt - leftpt) > epsilon);
   root = (rightpt+leftpt)/2;
   printf("\nRoot  is: %15.10lf\n", root);
   return 0;
}
```

If you take a calculus course you will learn another method for root finding called Newton's method