

Math 224: Scientific Computing I

Homework 2

Assigned Friday, Sept 3, 2004

Due Friday, Sept 10, 2004

Iterative Methods, Part 1: The Bisection Method

1. Reading: Atkinson 2.1, Trangenstein 5.2. Note: I think I forgot to include the appropriate updates to `fa`, `fb` in the code I described in lecture.
2. Write three C routines with the following declarations to carry out the bisection method to find a zero of the user-specified function $f(x)$ within the interval $a < x < b$:
 - (a) A version that stops after taking exactly n steps

```
float bisectf1(float (*f)(float x),float a,float b,int n);
```
 - (b) A version that stops when the error in the problem reaches a set tolerance, $|f(c)| \leq dy$

```
float bisectf2(float (*f)(float x),float a,float b,float dy);
```
 - (c) A version that stops when the error in the solution reaches a set tolerance, $|x_* - c| < dx$

```
float bisectf3(float (*f)(float x),float a,float b,float dx);
```

Explain how you estimate the value $|x_* - c|$.
 - (d) What does your code do if $f(a)f(b) > 0$? Have it do something, for example returning an “obviously-wrong” value for x_* to make it clear that something has gone wrong.
3. Write programs to apply the routines from (2) to find the zero of the function

```
float h(float x)
{
    return(tan(x/4.0)-1.0);
}
```

starting from $a = 0, b = 4$. The exact solution is $x = \pi$. To calculate the numerical error in your results, you may compare your answer against the C definition of π , called `M_PI`, for each of the following tests:

- (a) For `bisectf1()`, make a linear-log plot of the absolute value of the error in the solution as a function of the number of steps n for $n = 1, 2, 3, \dots, 200$.¹ What is your minimum achievable error in the solution? How does this relate to ϵ_{mach} ? How many steps n did it take? What happens if you take more steps than that?
- (b) For `bisectf2()`, create a global variable `int fcn_count`; to count the number of times that you use function `h()`.² Add the command `fcn_count++`; to `h()` before the `return()` to do this. Make a log-linear plot of the `fcn_count` value needed to reach a desired tolerance dy for $dy = 1, 0.5, 0.25, 2^{-3}, \dots, 2^{-100}$. What happens if dy is “too small”? What is the minimum value of dy that you can reach?
- (c) For `bisectf3()`, what is the minimum achievable error in the solution?

Write your most accurate solution.³

4. Based on your experiences from problems (2abcd,3), write your final version of the bisection method (using the stopping test of your choice):

```
double bisect(double (*f)(double x),double a,double b);
```

It should be designed to compute the solution as accurately as possible (given the value of ϵ_{mach} you calculated last week) and not do unnecessary calculations. You will make use of this code many times now and later in this course.

¹I recommend outputting your results to a file and reading that file into a graphing program: gnuplot, xmgr, matlab, ...

²Be sure to re-zero this each time you start another bisection.

³Be careful about how C's `printf` formatting commands to control the number of digits you print out.

5. James Wilkinson was a British numerical analyst. He suggested this problem, now called Wilkinson's polynomial⁴

$$W_\alpha(x) \equiv (x-1)(x-2)(x-3)\cdots(x-19)(x-20) - \alpha$$

- (a) Write a function `double w(double x)`; to calculate Wilkinson's polynomial. Declare `double alpha`; as a global variable in your program.
- (b) You will use your routine `bisect()` from (4) to find all of the real roots of Wilkinson's polynomial for various values of α , (i.e. find $W(x_*) = 0$). For $\alpha = 10^{11}$, you should be able to obtain one solution as $x_* \approx 8.9386518013726$. If you do not obtain a result very very close to this, you should revise your solution to (4) – Wilkinson's polynomial is regarded as one of the "ultimate tests" for any root-finding code.
- (c) To full accuracy (i.e. all digits), Write all the real roots for $\alpha = 10^{12}$.
This problem will be graded based on the number of digits of accuracy in your answers!
Hint: for "small" α the roots are close to the integers, $k = 1, 2, 3, \dots, 20$, so reasonable search brackets to try are $(k - 1/2, k + 1/2)$.
- (d) For α starting from $\alpha = 10^{10}$ up to $\alpha = 10^{20}$ going in steps of `alpha*=1.1`; or smaller, obtain all the real roots of Wilkinson's polynomial and save them to a file in two columns (first column is α , second is the root). Make a linear-log plot of these roots vs. α . Why are there less than twenty real roots for large α ? Explain what happens as α increases.

⁴A quote I found on the web stated; "In 1984, James H. Wilkinson admitted *Speaking for myself I regard it as the most traumatic experience in my career as a numerical analyst.*"