Introduction to Numerical Analysis

mz

19 June 2018

Write-along of the lecture notes by Amos Ron from the Computer Sciences department at Wisconson-Madison.

1. The Essence of Computation

What is computation? It isn't algebraic manipulation!

Fundamental limitations of numerical computation:

- *limited storage* - you can only store so much, so you cannot always perform exact computations, but only exact up to the limit of our precision e.g. 16 digits.
- *limited time* - computational operations take time, so we cannot do an infinite number of them.

So you want a method that can calculate a numerical solution that is accurate to within our limit, and takes as few operations as possible.

Calculate the square root of 5:

$$x_{new} = \frac{x_{old}^2 + 5}{2x_{old}} \tag{1}$$

```
# write function
FunSqrt <- function(x.old, i = 1) {
 x.new <- (x.old^2 + 5)/(2*x.old)
 i <<- i + 1
 print(paste("phase", i, "x_old = ", x.old,
             "x_new = ", x.new, "x_new^2 = ", x.new^2))
 x.old <<- x.new
}

# start with 2.2 as the first x_old
FunSqrt(2.2)
```

```
## [1] "phase 1 x_old =  2.2 x_new =  2.23636363636364 x_new^2 =  5.00132231404959"
```

```
FunSqrt(x.old)
```

```
## [1] "phase 1 x_old =  2.23636363636364 x_new =  2.23606799704361 x_new^2 =  5.00000008740261"
```

```
FunSqrt(x.old)
```

```
## [1] "phase 1 x_old =  2.23606799704361 x_new =  2.23606797749979 x_new^2 =  5"
```

So only after three runs of the function, each one completing four arithmetic operations, we have a solution accurate to the limit of our precision.

Lecture 2: Introduction Part II and Solving Equations

2.1 Numerical Solution to Quadratic Equations

So the quadratic euqation of the form

$x^2 + bx = c$

Has the familiar solutions

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 + 4c}}{2} \tag{2}$$

Evaluating this expression is computationally demanding because of the square root, the other operations are very fast, so not an issue. So the solution to the quadratic equation actually reduces to finding the squareroot. But is the algorighm for the square root better than the best algorithm for solving this equation without the formula? Turns out we can extend the square root algorithm to solve the quadratic equation better than the formula 2.

So last time we found the non-negative solution to the quadratic equation using an iterative method that allows us to control the precision of the solution. So we need

$$x^2 = c, x \geq 0, c \geq 0 \tag{3}$$

And we used equation 1 to iterate to the solution for a square root

2.2 Solving quadratic equations

So to solve equations of the form

$$x^2 + bx = c \tag{4}$$

We add the b term to the denominator:

$$x_{new} = \frac{x_{old}^2 + 5}{2x_{old} + b} \tag{5}$$

Now each iteration takes 5 operations (2 multiplications, 2 additions and 1 division) and we can again get arbitrarily precise. So the actual analytic solution to the quadratic equation was not necessary at all!

3. Calculating Definite Integrals

Another good example of the difference between numerical computation and analytical solutions.

Recal the definition of a definite integral of (positive) continuous funciton f(t) over $[a, b]$, $\int_a^b f(t)dt$ is the area under the curve.

Standard analytin solutin is to use the antiderivative, so find the function F such that $F' = f$, then:

$$\int_a^b f(t)dt = F(b) - F(a) \tag{6}$$

But that's only neat in some cases, in others you end up with a more conplicated solution. What we want depends on the funciton f, but the way to get it now depends on a new function F.

So how could we calculate the derivative using only information on f? Well, go back to the definition of the derivative: that of the area under the curve in order to approximate its value.

We split the interval $[a, b]$ into subinvervals of equals size h, then estimate the definite integral over each subinterval and add them all together. if we have N subintervals of equal width $h = \frac{(b-a)}{N}$, then the integral is

$$\int_a^b f(t)dt = \sum_{j=1}^{N} \int_{a+(j-1)h}^{a+jh} f(t)dt \tag{7}$$

Now all we've done is reduced the problem of estimating the area under the curve to a problem of estimating a series on N small definite integrals. These can be estimated using two strategies:

The Rectangle rule

Here the rectangle has the height $f(a)$, i.e. the left corner of the rectangle:

$$\int_{a+h}^{a} \approx f(a)h \tag{8}$$

The Midpoint rule Here you take the height at the midpont of the interval.

$$\int_{a+h}^{a} \approx f(a + h/2)h \tag{9}$$

You should always be able to evaluate f at any point, so this is simple, and you increase the precision by increasing the numnebr of intervals N. So why is our intuition that the midpoint rule would be better?

3.1 Evaluating Numeric Algorithms

Cost - the amount of time (e.g. number of operations) the computation will take. we can often tradeoff cost for accuracy.

Accuracy - the rate at which the error approaches zero

**Robustness* - How robust is the algorithm to different types of functions and inputs. e.g. the square root algorithm is super robust, even if we give it a terrible initial value.

3.1 Evaluation of Algorithms for Definite Integrals

So how do these criteria apply here:

- Cost: each algorighm on each subinterval costs the same, so the total cost grows linearly as the number of intervals N grows. So if you double N, the cost will double as well. There is therefore a fixed constant c_1 that is independent of N so that the total cost $c_1 * N$