Assignment C-programming

david redmond ID 14207377

30 July 2017

Objectives

The code implement 3 numerical root finding algorithms , Bisection method ,NewtonRaphson's method and Secant method

Methods of operation:

Bisection

This method is a simple root-finding algorithm for a function f being a continuous function, for which one knows an interval [a, b] such that f(a) and f(b) have opposite signs (a bracket). This is implemented simply by (f(a) * f(b) < 0) and one has divided by two the size of the interval. Each iteration performs these steps: 1. Calculate c, the midpoint of the interval, c = a + b/2. 1. Calculate the function value at the midpoint, f(c). 1. If convergence is satisfactory (that is, a - c is sufficiently small, or f(c) is sufficiently small), return c and stop iterating. 1. Examine the sign of f(c) and replace either (a, f(a)) or (b, f(b)) with (c, f(c)) so that there is a zero crossing within the new interval.

NewtonRaphson's

This method assumes the function f to have a continuous derivative. Newton's method may not converge if started too far away from a root. However, when it does converge, it is faster than the bisection method, and is usually quadratic. Newton's method is also important because it readily generalizes to higher-dimensional problems.

$$x(n+1) = x(n) - \frac{f'(x(n))}{f\ ''\ (x(n))}$$

Secant method

This method is replacing the derivative in Newton's method with a finite difference, we get the Secant method.

$$x[n] = x[n-1] - f(x[n-1]) * \frac{(x[n-1] - x[n-2])}{(f(x[n-1] - f(x[n-2]))}$$

Code structure :

Each algorithm is coded as per recurssion formula, and each functions will have a specific exit to the iteration based on the supplied tolerance value, or maximum number of iterations allowed, or if the exit condition is satisifed. The function also includes a check for convergence for Secant and Newton. The library incl, contains 4 functions : first and second derivatives log-likelihood for Cauchy PDF, and a min and max value function which is used for the initial guess in the algorithms

There is an exception handler **check_valid_inputs** that looks for valid inputs for number of iterations, and valid error tolerence.

Included is a sample test vector

```
xx <- c( 12.262307 , 10.281078 , 10.287090 , 12.734039 ,
         11.731881 , 8.861998 , 12.246509 , 11.244818 ,
         9.696278 , 11.557572 , 11.112531 , 10.550190 ,
         9.018438 , 10.704774 , 9.515617 , 10.003247 ,
         10.278352 , 9.709630 , 10.963905 , 17.314814)
```

Benchmarking

Given the following identical conditions each algorithm was benchmarked using the rbenchmark library

```
itr=200
tol = 1e-7
inita = 0.0
initb = 1.0
bisectC(xx, itr, tol, inita, initb)
inita = bisectC(xx, itr, tol, inita, initb)
secantC(xx, itr, tol, inita)
NewRapC(xx, itr, tol, inita)
#
benchmark(bisectC(xx, itr, tol, inita, initb),
          secantC(xx, itr, tol,inita),
          NewRapC(xx, itr, tol,inita),
          order = "relative")[,1:4]
```

Notice that the output of Bisect is used as the initial guess for Secant and Newton. This is because these 2 algorithms are not gauranteed to converge.

```
                                 test replications elapsed relative
2        secantC(xx, itr, tol, inita)          100   0.001        1
3        NewRapC(xx, itr, tol, inita)          100   0.001        1
1 bisectC(xx, itr, tol, inita, initb)          100   0.008        8
```

These results are as expected with the Bisect method taking much longer to converge then either the other methods. However this test is a little biased as the starting point for Newton, and Secant is closer to the root , otherwise these methods may not have converged.

Sensitivity of starting points

To evaluate how the functions performed using different start points i've modified the inita values and found that Secant in many cases failed to converge, similarly with the NewtonRaphson method.

Advantages and limitations.

BisectC

The method is guaranteed to converge to a root if f is a continuous function on the interval [a, b]. The absolute error is halved at each step so the method converges linearly, which is comparatively slow.

Newton's method

This is an extremely powerful technique—in general the convergence at a quadratic rate, however when there are many iterations, it's performance is slower because it requires the extra derivative be calculated and even using the **ll_2** function to accelerate that calculation. A large error in the initial estimate can contribute to non-convergence of this algorithm. ### Secant method Is very fast however it does not require a very good initial guess. when further benchmarking is performed with higher iteration, and smaller tolerences it is the fastest. However there is a higher risk of non-convergence.

Conclusion

Using a combination of the Bisect with few iterations and coarse tolerence to find an initial root, then using this root as the initial starting point for either Newton, or Secant to find the precision result.