3. User Defined R Functions

Cheng Peng

Lab Note for MAT325 Numerical Analysis

Contents

1 Introduction

One of the great strengths of R is that it allows users to extend the capacity of R through user-defined functions. Functions are often used to encapsulate a sequence of expressions that need to be executed repetitively. We have made code to implement bisection and fixed-point methods by specific examples. We may want to write the R function of these implementations so we can use them for other root-finding problems with the appropriate input information.

2 Syntax and Types of User-Defined R Functions

The syntax of the general user-defined R function has the following form

```
myfunction <- function(arg1, arg2, ... ){
   statements
   return(object)
}
```

`arg1, arg2, ...` are arguments that are passed into the function. The arguments could be any objects such as vectors and other user-defined R functions.

Example 1 Finding the standard deviation of an input vector.

```
sdfun <- function(x) {
  res <- sqrt(sum((x - mean(x))^2) / (length(x) - 1))
  return(res)
}
##
vec = c(1,4,2,6,-3, -5)
sdfun(x = vec)   # or simply sdfun(vec)
```

```
## [1] 4.167333
```

We have defined functions in implementing root-finding methods.

3 Writing R Functions for Root-finding Methods

As an example, we will write an R function to implement the bisection method for finding the root for any given equation on ver an interval (if it exists). Two versions of R functions will be presented in the following.

3.1 Function with Numerical Outputs

We simply wrap up the example code in the lecture note to make the following function.

```
num.FixedPoint = function(fn,        # function that satisfies f(x) = x
                          a,         # lower limit of the interval [a, b]
                          b,         # upper limit of the interval [a, b]
                          TOL,       # error tolerance
                          N,         # maximum number of iterations
                          x0,        # initial value of x
                          detail,     # intermediate output
                          ...){
 gfun = fn
 x = x0
 ERR = Inf
 n = 0
##
 while (ERR > TOL){
  n = n + 1
  new.x = gfun(x)
  ERR = abs(new.x - x)
  if(ERR < TOL){
    cat("\n\nThe algorithm converges!")
    cat("\nThe approximate root is:", new.x,".")
    cat("\nThe absolute error is:", ERR, ".")
    cat("\nThe number of iterations is:", n, ".")
    break
  } else{
    if(ERR > 10^7){
        cat("\n\nThe algorithm diverges!")
        break
    } else{
         if(detail == TRUE){
            cat("\nIteration:",n,". Estimated root:", new.x, ". Absolute error:", ERR,".")
          }
         x = new.x         # update x value!!!
    }
  }
  if(n == N){
    cat("\n\nThe maximum number of iterations is achieved!")
    break
  }
 }
}
```

Next, we use several examples.

Example 1 Using the fixed-point method to find the approximate root of $x^3 - 7x + 2 = 0$. To use the fixed-point method, we rewrite the equation into the form $(x^3 + 2)/7 = x$. Then $g(x) = (x^3 + 2)/7$ will be the function to be passed into the function.

```
###
fun0 = function(x) (x^3 + 2)/7
num.FixedPoint(fn = fun0, a = 0, b = 2, TOL = 10^(-6), N = 200, x0 =1.5, detail = TRUE)
```

```
##
## Iteration: 1 . Estimated root: 0.7678571 . Absolute error: 0.7321429 .
## Iteration: 2 . Estimated root: 0.3503903 . Absolute error: 0.4174668 .
## Iteration: 3 . Estimated root: 0.2918598 . Absolute error: 0.0585305 .
## Iteration: 4 . Estimated root: 0.2892659 . Absolute error: 0.002593907 .
## Iteration: 5 . Estimated root: 0.289172 . Absolute error: 9.385572e-05 .
## Iteration: 6 . Estimated root: 0.2891687 . Absolute error: 3.364631e-06 .
##
## The algorithm converges!
## The approximate root is: 0.2891686 .
## The absolute error is: 1.20578e-07 .
## The number of iterations is: 7 .
```

Example 2: Calculate $1/\sqrt{2}$ by using the fixed-point method. Note that $f(x) = x^2 - 1/2$. To use the fixed point method, we need $g(x) = x - x^2 + 1/2$ as the input function.

```
###
fun0 = function(x) x - x^3 + 1/2
num.FixedPoint(fn = fun0, a = -1, b = 2, TOL = 10^(-5), N = 200, x0 =0.7, detail =FALSE)
```

```
##
##
## The algorithm converges!
## The approximate root is: 0.7937048 .
## The absolute error is: 9.128962e-06 .
## The number of iterations is: 83 .
```

3.2 Function with More Optional Outputs

We include graphics the function created previously.

```
FixedPointAlgR = function(fn,       # function that satisfies f(x) = x
                          a,        # lower limit of the interval [a, b]
                          b,        # upper limit of the interval [a, b]
                          TOL,      # error tolerance
                          N,        # maximum number of iterations
                          x0,       # initial value of x
                          detail,    # intermediate output
                          graphic=TRUE,
                          x.lim,
                          y.lim,
                          sleep,
                          ...){
 gfun = fn
 x = x0
 ERR = Inf
 n = 0
```

```r
##
 if(graphic==TRUE){
 xlimit=c(a-0.1*abs(b-a), b+0.1*abs(b-a))
 xx=seq(a-0.1*abs(b-a), b+0.1*abs(b-a), length = 2000)
 yy1 = gfun(xx)
 yy2 = xx
 plot(xx, yy1, ylim = y.lim, xlim = x.lim, type = "l", lwd = 2, lty = 1, col = "blue")
 lines(xx,yy2, lwd = 2, lty = 1, col ="darkred")
 title("Fixed Point Algorithm Approximation")
 }
##
 while (ERR > TOL){
  Sys.sleep(sleep)
  n = n + 1
  new.x = gfun(x)
  ERR = abs(new.x - x)
  if(ERR < TOL){
    cat("\n\nThe algorithm converges!")
    cat("\nThe approximate root is:", new.x,".")
    cat("\nThe absolute error is:", ERR, ".")
    cat("\nThe number of iterations is:", n, ".")
    break
  } else{
    if(ERR > 10^7){
        cat("\n\nThe algorithm diverges!")
        break
    } else{
         if(detail == TRUE){
            cat("\nIteration:",n,". Estimated root:", new.x, ". Absolute error:", ERR,".")
         }
        if(graphic==TRUE){
            segments(new.x, new.x,       new.x,       gfun(new.x), col="purple")
            segments(new.x, gfun(new.x), gfun(new.x), gfun(new.x), col="purple")
          }

         x = new.x         # update x value!!!
    }
  }
  if(n == N){
    cat("\n\nThe maximum number of iterations is achieved!")
    break
  }
 }
}
```

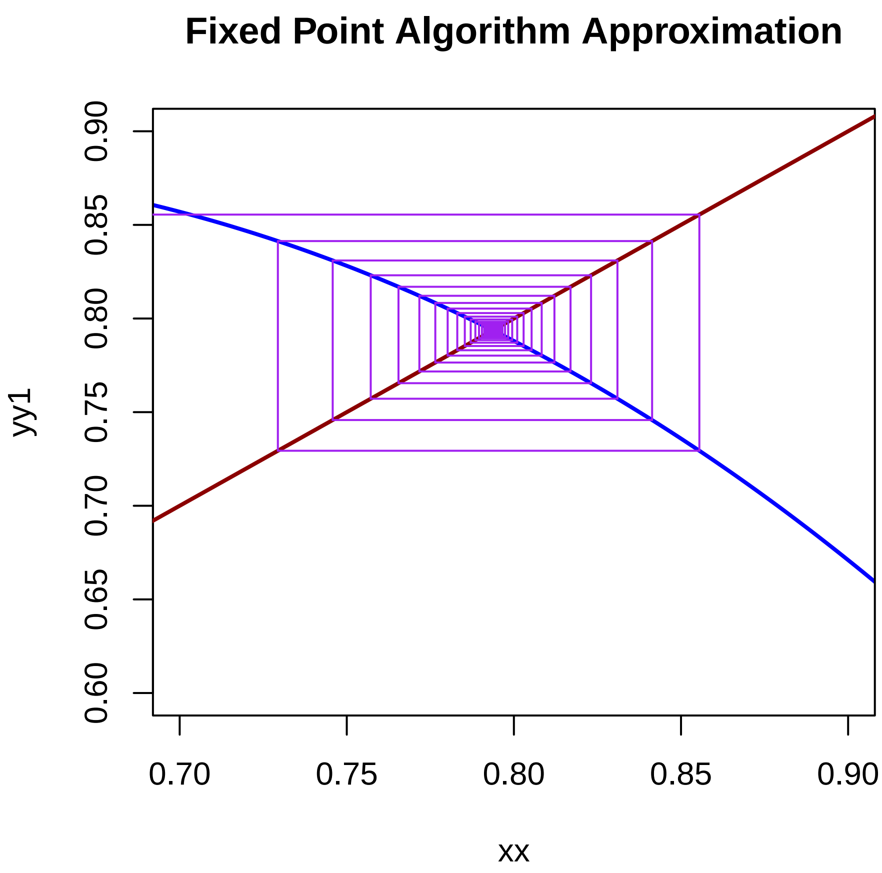
Example 1 (Revisited)

```r
###
fun0 = function(x) (x^3 + 2)/7
FixedPointAlgR(fn = fun0, a = 0, b = 1, TOL = 10^(-6), N = 200,
               x0 =0.35, detail = TRUE, graphic = FALSE,
               x.lim=c(0.285, 0.294), y.lim=c(0.28, 0.32), sleep = 0.1)
```

```
##
## Iteration: 1 . Estimated root: 0.2918393 . Absolute error: 0.05816071 .
## Iteration: 2 . Estimated root: 0.2892651 . Absolute error: 0.002574143 .
## Iteration: 3 . Estimated root: 0.289172 . Absolute error: 9.313374e-05 .
## Iteration: 4 . Estimated root: 0.2891687 . Absolute error: 3.33874e-06 .
##
## The algorithm converges!
## The approximate root is: 0.2891686 .
## The absolute error is: 1.196502e-07 .
## The number of iterations is: 5 .
```

Example 2 (Revisited)

```
###
fun0 = function(x) x - x^3 + 1/2
FixedPointAlgR(fn = fun0, a = -1, b = 2, TOL = 10^(-5), N = 200,
               x0 =-1.2, detail =FALSE, graphic = TRUE,
               x.lim=c(0.7, 0.9), y.lim=c(0.6, 0.9), sleep = 0.1)
```




```
##
##
## The algorithm converges!
```

```
## The approximate root is: 0.7937047 .
## The absolute error is: 8.942928e-06 .
## The number of iterations is: 85 .
```