Roots of Equation

Exercise

1. *Required Competencies*: False Position Method, functions, loops
   Create an R function named **FalsePositionMethod** with the following variables as parameters:

- **f,** a variable which handles an arbitrary mathematical function;
- **a**, a numerical variable for the lower limit of the interval;
- **b**, a numerical variable for the upper limit of the interval;
- **macheps**, a numerical variable for the machine epsilon;
- **max**, a numerical variable for the maximum number of iterations, with 1000 as default; and
- **verbose**, an optional Boolean value if the whole process is to be printed. Default value is TRUE.

The function should perform the prescribed method operating on the above variables, and must return a list which contains the values of the following variables:

- **f**
- **given_a**
- **given_b**
- **c**, the converged value
- **iterations**, the iteration count when the method converged
- **ea**, the approximate error of the last iteration.

If the verbose variable is set as TRUE, the function should print the values of a, b, c and their respective evaluations over the function f, as well as the approximate error for c, for each iteration.

2. *Required Competencies*: Secant Method, functions, loops
   Create an R function named **SecantMethod** with the following variables as parameters:

- **f**, a variable which handles an arbitrary mathematical function;
- **x0**, a numerical variable for the first root approximate;
- **x1**, a numerical variable for the second root approximate;
- **macheps**, a numerical variable for the machine epsilon;
- **max**, a numerical variable for the maximum number of iterations; and
- **verbose**, an optional Boolean value if the whole process is to be printed. Default value is TRUE.

The function should perform the prescribed method operating on the above variables, and must return a list which contains the values of the following variables:

- **f**
- **given_x0**
- **given_x1**
- **x**, the current value of x at the current iteration
- **iterations**, the current iteration count
- **ea**, the approximate error of the last iteration.

If the verbose variable is set as TRUE, the function should print the values of x0, x1, x and their respective evaluations over the function f, as well as the approximate error for x, for each iteration.

3. *Required Competencies*: Muller's Method, functions, loops
   Create an R function named **MullerMethod** with the following variables as parameters:

- **f**, a variable which handles an arbitrary mathematical function;

- **x0**, a numerical variable for the first root approximate;
- **x1**, a numerical variable for the second root approximate;
- **x2**, a numerical variable for the third root approximate;
- **macheps**, a numerical variable for the machine epsilon;
- **max**, a numerical variable for the maximum number of iterations; and
- **verbose**, an optional Boolean value if the whole process is to be printed. Default value is TRUE.

The function should perform the prescribed method operating on the above variables, and must return a list which contains the values of the following variables:

- **f**
- **given_x0**
- **given_x1**
- **given_x2**
- **x3**, the current value of x3 at the current iteration
- **iterations**, the current iteration count
- **ea**, the approximate error of the last iteration.

If the verbose variable is set as TRUE, the function should print the values of x0, x1, x2, and x3 and their respective evaluations over the function f, as well as the approximate error for x3, for each iteration.

Test Cases

```
> f = function(x) -26 + 85*x - 91*x^2 - 44*x^3 - 8*x^4 + x^5
> FalsePositionMethod(f, 10, 15, 1e-5, 1000, TRUE)
  iteration     a  b       f(a)   f(b)     c       f(c)        ea
 [1,]         1 10.00 15 -2.769e+04 186649 10.74 -2.769e+04 1.000e+02
 [2,]         2 10.74 15 -2.057e+04 186649 11.29 -2.057e+04 4.879e+00
 [3,]         3 11.29 15 -1.359e+04 186649 11.66 -1.359e+04 3.161e+00
 [4,]         4 11.66 15 -8.267e+03 186649 11.88 -8.267e+03 1.909e+00
 [5,]         5 11.88 15 -4.776e+03 186649 12.02 -4.776e+03 1.100e+00
 [6,]         6 12.02 15 -2.676e+03 186649 12.09 -2.676e+03 6.160e-01
 [7,]         7 12.09 15 -1.474e+03 186649 12.13 -1.474e+03 3.391e-01
 [8,]         8 12.13 15 -8.036e+02 186649 12.15 -8.036e+02 1.849e-01
 [9,]         9 12.15 15 -4.359e+02 186649 12.17 -4.359e+02 1.003e-01
[10,]        10 12.17 15 -2.358e+02 186649 12.17 -2.358e+02 5.426e-02
[11,]        11 12.17 15 -1.274e+02 186649 12.18 -1.274e+02 2.930e-02
[12,]        12 12.18 15 -6.872e+01 186649 12.18 -6.872e+01 1.581e-02
[13,]        13 12.18 15 -3.707e+01 186649 12.18 -3.707e+01 8.529e-03
[14,]        14 12.18 15 -1.999e+01 186649 12.18 -1.999e+01 4.599e-03
[15,]        15 12.18 15 -1.078e+01 186649 12.18 -1.078e+01 2.480e-03
[16,]        16 12.18 15 -5.811e+00 186649 12.18 -5.811e+00 1.337e-03
[17,]        17 12.18 15 -3.133e+00 186649 12.18 -3.133e+00 7.208e-04
[18,]        18 12.18 15 -1.689e+00 186649 12.18 -1.689e+00 3.886e-04
[19,]        19 12.18 15 -9.105e-01 186649 12.18 -9.105e-01 2.095e-04
[20,]        20 12.18 15 -4.908e-01 186649 12.18 -4.908e-01 1.129e-04
[21,]        21 12.18 15 -2.646e-01 186649 12.18 -2.646e-01 6.088e-05
[22,]        22 12.18 15 -1.426e-01 186649 12.18 -1.426e-01 3.282e-05
[23,]        23 12.18 15 -7.690e-02 186649 12.18 -7.690e-02 1.769e-05
[24,]        24 12.18 15 -4.146e-02 186649 12.18 -4.146e-02 9.539e-06
$`f`
```

```
function (x)
-26 + 85 * x - 91 * x^2 - 44 * x^3 - 8 * x^4 + x^5
<bytecode: 0x000000000ea723f0>
$given_a
[1] 10
$given_b
[1] 15
$c
[1] 12.18
$iterations
[1] 24
$ea
[1] 9.539e-06

> SecantMethod(f, 10, 15, 1e-5, 1000, TRUE)
      iteration    x0    x1      f(x0)      f(x1)     x       f(x)        ea
 [1,]         1 10.00 15.00 -3.228e+04  1.866e+05 10.74 -2.769e+04 1.000e+02
 [2,]         2 15.00 10.74  1.866e+05 -2.769e+04 11.29 -2.057e+04 4.879e+00
 [3,]         3 10.74 11.29 -2.769e+04 -2.057e+04 12.88  2.616e+04 1.235e+01
 [4,]         4 11.29 12.88 -2.057e+04  2.616e+04 11.99 -5.533e+03 7.426e+00
 [5,]         5 12.88 11.99  2.616e+04 -5.533e+03 12.14 -1.115e+03 1.280e+00
 [6,]         6 11.99 12.14 -5.533e+03 -1.115e+03 12.18  6.876e+01 3.218e-01
 [7,]         7 12.14 12.18 -1.115e+03  6.876e+01 12.18 -7.749e-01 1.870e-02
 [8,]         8 12.18 12.18  6.876e+01 -7.749e-01 12.18 -5.290e-04 2.084e-04
 [9,]         9 12.18 12.18 -7.749e-01 -5.290e-04 12.18  4.075e-09 1.424e-07
$`f`
function (x)
-26 + 85 * x - 91 * x^2 - 44 * x^3 - 8 * x^4 + x^5
<bytecode: 0x000000000ea723f0>
$given_x0
[1] 10
$given_x1
[1] 15
$x
[1] 12.18
$iterations
[1] 9
$ea
[1] 1.424e-07

> MullerMethod(f, 8, 10, 15, 1e-5, 1000, TRUE)
     i    x0    x1    x2    f(x0)      f(x1)      f(x2)     A     B          C    x3
[1,] 1  8.00 10.00 15.00 -27698.0 -3.228e+04  1.866e+05  6582 76695  1.866e+05 11.54
[2,] 2 10.00 15.00 11.54 -32276.0  1.866e+05 -1.605e+04  9598 25307 -1.605e+04 12.07
[3,] 3 15.00 11.54 12.07 186649.0 -1.605e+04 -3.364e+03 11767 30232 -3.364e+03 12.17
[4,] 4 11.54 12.07 12.17 -16054.1 -3.364e+03 -2.235e+02  8474 30302 -2.235e+02 12.18
[5,] 5 12.07 12.17 12.18  -3363.9 -2.235e+02  5.455e-01  9123 30506  5.455e-01 12.18
[6,] 6 12.17 12.18 12.18   -223.5  5.455e-01 -1.564e-05  9242 30506 -1.564e-05 12.18
          f(x3)        ea
[1,] -1.605e+04 3.001e+01
[2,] -3.364e+03 4.380e+00
```

```
[3,] -2.235e+02 8.776e-01
[4,]  5.455e-01 6.043e-02
[5,] -1.564e-05 1.468e-04
[6,] -5.821e-11 4.209e-09
$`f`
function (x)
-26 + 85 * x - 91 * x^2 - 44 * x^3 - 8 * x^4 + x^5
<bytecode: 0x000000000ea723f0>
$given_x0
[1] 8
$given_x1
[1] 10
$given_x2
[1] 15
$x3
[1] 12.18
$iterations
[1] 6
$ea
[1] 4.209e-09
```

Word Problem

You are designing a spherical tank (see figure below) to hold water for a small village in a developing country. The volume of liquid it can hold can be computed as:

$$V = \pi h^2 \frac{(3R-h)}{3}$$

where $V = volume\ (m^3), h = depth\ of\ water\ in\ tank\ (m)$, and $R = tank\ radius\ (m)$. If $R = 3m$, what depth must the tank be filled to so that it holds $30m^3$? Solve this problem using any of the root finding methods discussed in the laboratory. Perform a maximum of five iterations of your chosen method and write in a one-half crosswise the derived function from the problem and the different values and relative error per iteration. You can use your code to verify your answer. (Hint: h must be a non-negative value and it must not exceed the diameter of the sphere).