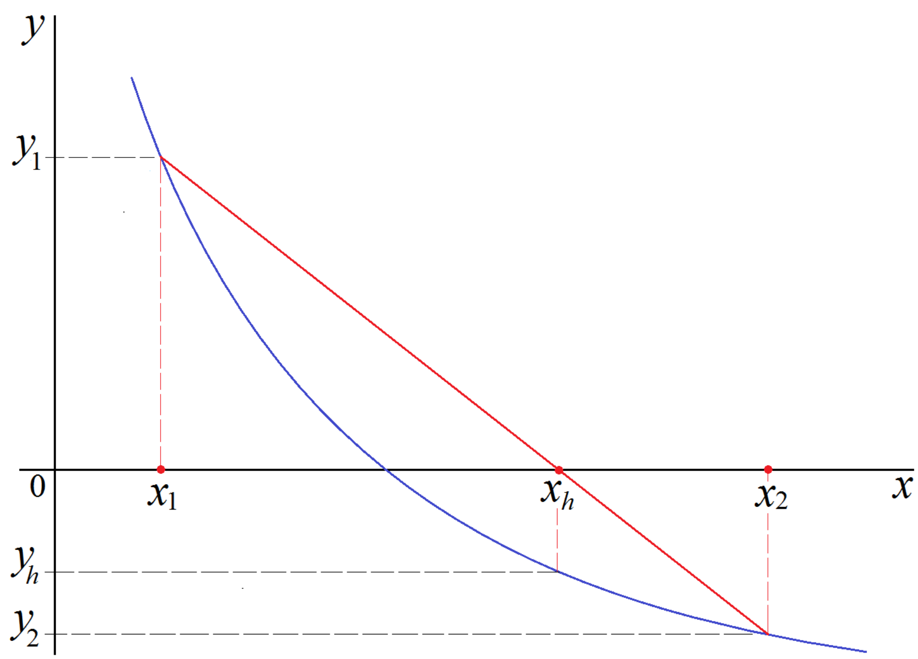
False Position (Regula Falsi) Method

This method is very similar in algorithm to the bisection method since it requires two initial values of x that should (*i*) embrace the location of the expected root, and (*ii*) have $y(x)$ of different signs. The difference is the method of calculating the next values of x.

In false position method, the position of x_h is the intersection point of the x-axis and the line connecting the two *false* points (x_1, y_1) and (x_2, y_2), as shown in the figure below.



Initial values of the false position method

By using the slope of the line, the relation between the three points can be established as

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y_2 - 0}{x_2 - x_h}$$

So,

$$x_h = x_2 - \frac{x_2 - x_1}{y_2 - y_1} y_2$$

According to the sign of y_h, the x_h is renamed as x_1 or x_2 in such a way that always keeps the root between x_1 and x_2. This can be achieved by keeping x_1 and x_2 at the opposite sides of x-axis. The procedure continues until the value of y_h approaches zero.

Example 4.5

Solve the following equation:

$$2x^2 - 5x + 3 = 0$$

Solution

Let's write the equation as a function

$$y(x) = 2x^2 - 5x + 3$$

The same algorithm of the bisection method is used. In this example, the function `rflasi()` is constructed to solve the equation.

```
def y(x): return 2*x**2 - 5*x + 3

def rfalsi(fn, x1, x2, tol = 1.0e-6, ilimit = 100):
    y1 = y(x1)
    y2 = y(x2)
    xh = 0
    ipos = 0                 # counts false positions
    if y1 == 0: xh = x1      # if x1 is a root
    elif y2 == 0: xh = x2    # if x2 is a root
    elif y1 * y2 > 0:        # if y1 and y2 have the same sign
        print('No root exists within the given interval.')
    else:
        for ipos in range(1,ilimit+1):
            xh = x2 - (x2-x1)/(y2-y1) * y2
            yh = y(xh)
            if abs(yh) < tol:
                break
            elif y1 * yh < 0: # if y1 and yh have opposite signs
                x2 = xh
                y2 = yh
            else:
                x1 = xh
                y1 = yh
    return xh, ipos

x1 = float(input('Enter the value x1: '))
x2 = float(input('Enter the value x2: '))
x, n = rfalsi(y,x1,x2)
# Output of results
print('The root: %.5f' % x)
print('The number of computed false positions: %d' % n)
```

The code is executed as following

```
Enter the value x1: 0
Enter the value x2: 1.2
The root: 1.00000
The number of computed false positions: 32
```

Let's enter a smaller interval around the expected solution:

```
Enter the value x1: .5
Enter the value x2: 1.1
The root: 1.00000
The number of computed false positions: 17
```

Reversing the input sequence of the limits of the initial interval can be tested:

```
Enter the value x1: 1.2
Enter the value x2: 0
The root: 1.00000
The number of computed false positions: 32
```

The second root can be found by entering other two values.

```
Enter the value x1: 1.2
Enter the value x2: 3
The root: 1.50000
The number of computed false positions: 48
```