Python code and analysis

```python
#!/usr/bin/python

import numpy as np

def trapezoidal(function,a,b,stepsize):
    """
    Function that takes a function, lower limit a, upper limit b and stepsize and calculates the integral using trapezoidal method.

    """
    interval = np.arange(a,b+stepsize,stepsize) ## array of the interval points
    integral = 0 ## variable to store the sum
    for i in range(1,len(interval)-1,2):
        integral += (stepsize/2)*(function(interval[i-1]) + 2*function(interval[i]) + function(interval[i+1]))
    return integral

def simpsons(function,a,b,stepsize):
    """
    Function that takes a function, lower limit a, upper limit b and stepsize and calculates the integral using simpsons method.
    """
    interval = np.arange(a,b+stepsize,stepsize) ## array of interrval points
    integral = 0 ## variable to store the sum
    for i in range(1,len(interval)-1,2):
        integral += (stepsize/3)*(function(interval[i-1]) + 4*function(interval[i]) + function(interval[i+1]))
    return integral

######### Define your function and limits of integration ############

def func_l(u):
    return 3*(1 - (u)**3)**(-1/3)

def func_r(s):
    return (3/2)*(1 - (s)**(3/2))**(-2/3)

a_l = 0
b_l = (0.5)**(1/3)
no_of_bins = 10**3
stepsize_l = (b_l - a_l)/no_of_bins

a_r = 0
b_r = (0.5)**(2/3)
stepsize_r = (b_r - a_r)/no_of_bins

# The final answer in trapezoidal and simpsons methods respectively are
t_inte = trapezoidal(func_l,a_l,b_l,stepsize_l) + trapezoidal(func_r,a_r,b_r,stepsize_r)
s_inte = simpsons(func_l,a_l,b_l,stepsize_l) + simpsons(func_r,a_r,b_r,stepsize_r)
```

The outputs are shown below and the values that already converge are shaded

No. of bins	Step size (left)	Step size (right)	Trapezoidal	$\frac{Trapezoidal}{Theoretical}$	Simpsons	$\frac{Simpsons}{Theoretical}$
10	0.079370	0.062996	3.631300	1.001020	3.627696	1.000026
100	0.007937	0.006300	3.627636	1.000010	3.627599	1.000000
1000	0.000794	0.000630	3.627599	1.000000	3.627599	1.000000
10000	7.93700e-05	6.29960e-05	3.627599	1.000000	3.627599	1.000000

C++ code and analysis

```cpp
#include <iostream>
#include <math.h>
using namespace std;

double f_l(double u){
        double e = -1.0/3.0;
        return 3.0*pow((1 - pow(u,3.0)),e);
}

double f_r(double s){
        double g = 3.0/2.0;
        double h = -2.0/3.0;
        return g*pow((1 - pow(s,g)),h);

}

int main(){
        double a_l = 0.0;
        double b_l = pow(0.5,1.0/3.0);
        double number_of_steps = pow(10,4);
        double stepsize_l = (b_l - a_l)/number_of_steps;

        double trap_l = 0;
        double simp_l = 0;
        for (double i = a_l + stepsize_l; i < b_l; i += 2.0*stepsize_l){
                trap_l += (stepsize_l/2.0)*(f_l(i-stepsize_l) + 2.0*f_l(i) + f_l(i+stepsize_l));
                simp_l += (stepsize_l/3.0)*(f_l(i-stepsize_l) + 4.0*f_l(i) + f_l(i+stepsize_l));
        }


        double a_r = 0.0;
        double b_r = pow(0.5,2.0/3.0);
        double stepsize_r = (b_r - a_r)/number_of_steps;

        double trap_r = 0;
        double simp_r = 0;
        for (double i = a_r + stepsize_r; i < b_r; i += 2.0*stepsize_r){
                trap_r += (stepsize_r/2.0)*(f_r(i-stepsize_r) + 2.0*f_r(i) + f_r(i+stepsize_r));
                simp_r += (stepsize_r/3.0)*(f_r(i-stepsize_r) + 4.0*f_r(i) + f_r(i+stepsize_r));
        }

        double final_trap = trap_l + trap_r;
        double final_simp = simp_l + simp_r;

        cout << "Step size : (left) : " << stepsize_l << " (right) : "<< stepsize_r << endl;
        cout << "Trapezoidal method : " << final_trap << endl;
        cout << "Simpsons method : " << final_simp << endl;
        return 0;
}
```

The results are :

No. of bins	Step size (left)	Step size (right)	Trapezoidal	$\frac{Trapezoidal}{Theoretical}$	Simpsons	$\frac{Simpsons}{Theoretical}$
10	0.079370	0.062996	3.6313	1.0010	3.6277	1.0000
100	0.007937	0.006300	3.62764	1.0000	3.6276	1.0000
1000	0.000794	0.000630	3.6276	1.0000	3.6276	1.0000
10000	7.93700e-05	6.29960e-05	3.6276	1.0000	3.6276	1.0000