67 lines (53 sloc) 1.66 KB

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <assert.h>

#define PI ( 3.1415926535 )

char is_even(int value) {  // Retorna se o número é par
    return value % 2 == 0;
}

char is_odd(int value) { // Retorna se o número é ímpar
    return !is_even(value);
}

double f(double t) {  // função f(x)
    return 1.0/sqrt(2 * PI) * exp(-(t*t)/2);
}

double int_f(double from, double to, int points) {  // ∫f(x)  Pelo método de Simpson
    assert(is_odd(points));
    double h = (to - from)/(points - 1);
    double acc = 0;
    acc += f(from);
    for (int i = 1; i < points - 1; i++) {
        int mult = is_even(i) ? 2 : 4;
        acc += mult * f(i * h + from);
    }
    acc += f(to);

    return h/3.0 * acc;
}

double F(double x) {  // equação proposta
    return int_f(0.0, x, 13) - 0.45;
}

double newton_root(  // método de newton
        double (*f)(double), // função para achar a raiz
        double (*f_prime)(double), // funcão derivada
        double x0, // ponto inicial
        double EPS /* Tolerância*/) {

    double eps = 10;
    double x = x0;
    while (eps >= EPS) {
        double old_x = x;
        x = x - f(x)/f_prime(x);
        eps = fabs(x - old_x);
    }

    return x;
}

int main(int argc, char *argv[]) {

    printf("Sendo F(1) = %lf\n", F(1));
    printf("Sendo F(2) = %lf\n", F(2));
    printf("F(1) * F(2) = %lf < 0\n", F(1) * F(2));
    puts("Portanto F possui raiz no intervalo [1, 2]");
    double root = newton_root(F, f, 0.5, 1e-10);
    printf("Pelo metodo de newton, a raiz de F no intervalo ocorre em x = %.10lf\n", root);
    printf("Conferindo o resultado: F(%lf) = %lf\n", root, F(root));

    return EXIT_SUCCESS;
}
```