CS 342
Project 1

Ata Deniz Aydın
21502637

The programs ./integral and ./tintegral were executed with the parameters L=5, U=105, K=1000 and numerous values of N, for numerous functions, and timed using the time command. The times each call took, in milliseconds, are tabulated in the tables below.

N	1	5	10	20	50	100
f(x) = x	1	2	3	7	31	53
f(x) = x^2	2	3	7	24	38	52
f(x) = 1/x	2	2	7	9	40	54
f(x) = e^x	2	3	17	23	43	89
f(x) = sin(x)	2	2	4	24	42	96

Figure 1. Total running times of ./integral 5 105 1000 N with the given functions.

N	10	20	50	100	200	500
f(x) = x	2	5	10	23	23	64
f(x) = x^2	2	4	10	27	43	60
f(x) = 1/x	2	4	4	25	28	73
f(x) = e^x	3	18	24	56	91	210
f(x) = sin(x)	3	17	25	54	89	188

Figure 2. Total running times of ./tintegral 5 105 1000 N with the given functions.

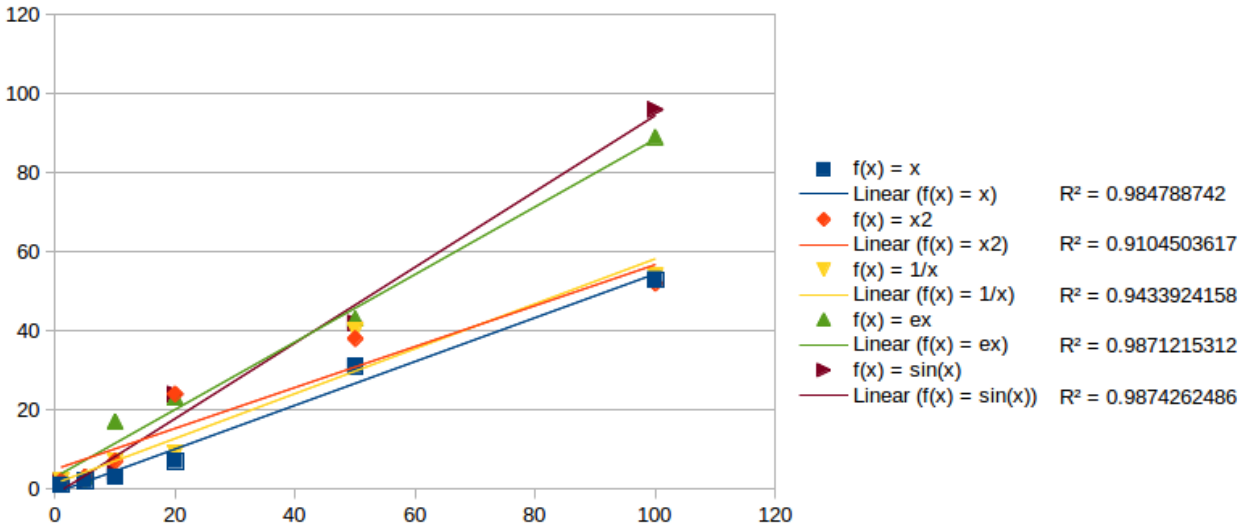

Figure 3. Scatter plot of running times of integral for different values of N and different functions, along with linear trend lines obtained via linear regression, with coefficients of determination included in the legend.

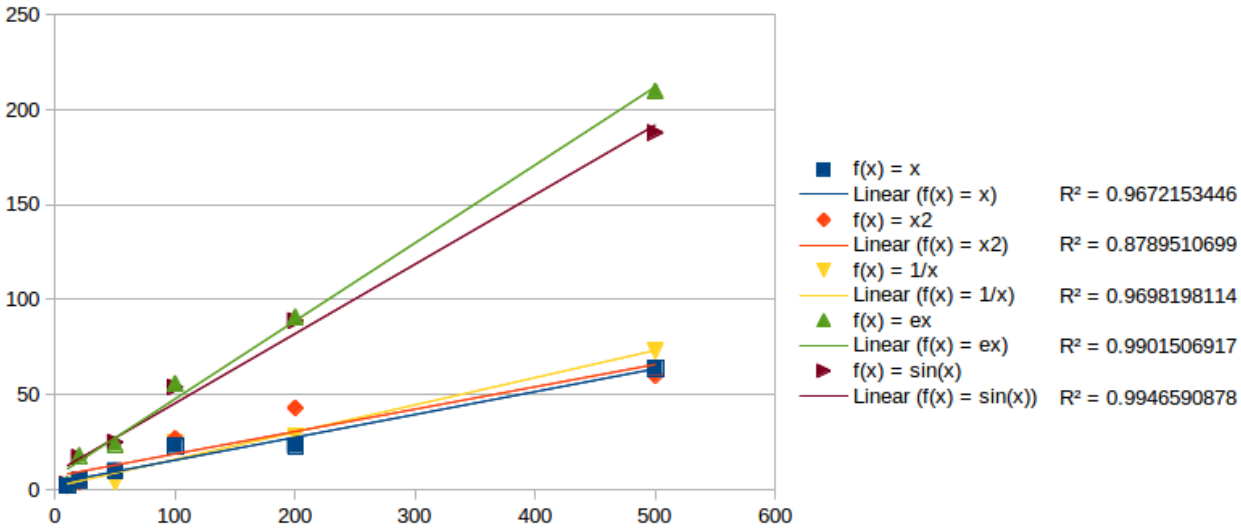

Figure 4. Scatter plot of running times of tintegral for different values of N and different functions, along with linear trend lines and their coefficients of determination.

From these figures, it can be observed that the runtime of the program increases approximately linearly with the number of processes or threads, and thread creation is less costly than process creation. For the same number of threads and processes, tintegral executed twice, and sometimes three times as fast as integral. We can also observe that library calls to math.h, as in e^x and sin(x), may increase the runtime of the program as much as 130%. Moreover, for small values of N, we may sometimes observe the regression line undershooting the actual data, which may be caused by a constant overhead in kernel operations or in the compiled program.