

Data plotting techniques for CS 475 & GRAD 510

In empirical computer science, especially in the context of this class, you will plot functions that have some common properties.

One class of functions describe the running time of a program or algorithm as a function of the input size. These functions are (i) positive, (ii) monotonically increasing and (iii) have a simple formula, especially when we consider the asymptotic behavior as the independent variable increases. We usually encounter three families of such functions—logarithmic, polynomial (linear functions are an important special case), and exponential.

Another important class consists of functions that are (i) positive, (ii) monotonically decreasing, and (iii) are also described by some common formulas. Typically, they represent the running time of programs as a function of the number of processors used. Regardless of the nature of the function, the cardinal rule for plotting data is:

Visually, the most information is conveyed by a straight line. Even if your function is not a straight line, it is often very useful to “massage” the data so that what is plotted comes through as a straight line.

We will now see numerous examples and tricks that illustrate this point.

- Consider a linear function, $y = ax + b$, for $a \geq 0$. Clearly, the data will appear as a straight line, monotonically increasing as x increases. The *slope* and *offset* of the line can be directly “read off the figure.” They provide a visual indication/confirmation of the values of the two defining parameters of the function, namely a and b .
- Now consider a polynomial function with a single monomial term, $y = ax^b$. The most important parameter of this function is b the degree of the polynomial, and the second important parameter is a the coefficient. If we directly plot this function, all we will see is a curve that is increasing more “sharply” than a straight line. It is visually very difficult to gauge the two parameters from the plot. Let us try to “massage” this function so that we get a straight line. Define

$y' = \log y$. Then $y' = \log a + b \log x$. If we define $x' = \log x$ and $\alpha = \log a$, we get $y' = bx' + \alpha$, which is a straight line. In other words, if we plot our data on a log-log scale, we should get a straight line. Moreover the slope of the line directly tells us the degree of the polynomial, and the offset gives the (logarithm of) the coefficient a .

Now what about a more complicated polynomial which has more than one term, such as $y = a_1x^{b_1} + a_2x^{b_2}$ for $b_1 > b_2$ (note that $y = ax^b + c$ is a particular case of this). Direct application of our logarithm trick does not work—we don't have a simple formula for the log of a sum of two terms. However, if we are dealing with the running time of programs as the input size grows, we know from complexity theory that the monomial with the largest degree dominates asymptotically. So we approximate the function as $y \approx a_1x^{b_1}$. In practical terms, we look at the log-log plot of our data, and if we see that the trailing part of our data is a straight line, we can be confident that the data represents a polynomial.

- Now consider an exponential function $y = ab^x$. Again the two important parameters are the growth rate b and the coefficient a . A direct plot of the function just shows a sharply increasing curve but it is difficult to judge whether it is a polynomial of high degree or an exponential function. However, if we first look at the log-log plot of our data and see that even this is increasing much more sharply than a straight line, we have reason to suspect that the function is exponential. To confirm this, we again look at the logarithm of y , i.e., $y' = \log y = \log a + x \log b$, thus y' grows linearly with x . In other words, a *semi-log* plot of the data will appear as a straight line.
- The second class of functions we will encounter represent the running time of parallel programs as a function of the number of processors. We expect these functions to decrease *hyperbolically* with the independent variable, i.e., $f(x) = \frac{c}{x}$. Here, note that $f(1) = c$, and represents the running time on a single processor. Again, a simple plot of the raw data is a curve that asymptotically approaches the x axis, but it is difficult to visually get any information from such a curve, especially if one is comparing two different implementations. We can use our log-log technique (after all this is a polynomial of degree -1 , so we will get a decreasing straight line with a 45° downward slope) but there is a better trick, which also extends to the more realistic case. We simply plot the reciprocal of the function, i.e., we define $y' = \frac{1}{f(x)} = \frac{x}{c}$. This is a straight line with slope $\frac{1}{c}$.

Sometimes we may scale it by c , i.e., $y' = \frac{f(1)}{f(x)} = \frac{c}{f(x)} = x$. This quantity is called the speedup, and is a straight line with unit slope.

Now consider the non-ideal case when the function is still monotonically decreasing, but has the form $f(x) = a + \frac{c}{x}$ where the function asymptotically approaches a constant, a . In this case, it is still useful to plot the speedup, i.e., the ratio $\frac{f(1)}{f(x)}$. Usually, this is not a straight line, but a curve that starts growing linearly, but flattens out to a constant.

You will find these techniques useful when you do your first homework, HW0.