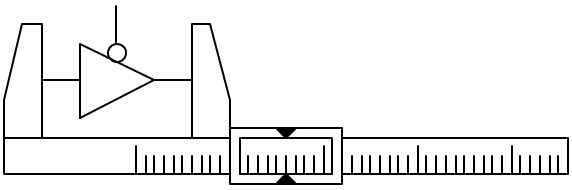


Ack User Guide

Accuracy figure of merit calculator
A C program for quantifying the correlation between two waveforms

Revision 1.1
October 23, 2001
Greg Edlund
gedlund@us.ibm.com

Ack is a program that computes figures of merit for the correlation of two waveforms in the *voltage* domain. (Time domain correlation would involve establishing a time reference using a standard load; this would probably be a simpler program.) Ack is a general-purpose program and should work equally well with any two x-y waveforms. The figures of merit and algorithm grew out of the "I/O Buffer Accuracy Handbook." (See references.) Incidentally, the name is inspired by Bill the Cat, a flea-bitten bag of fur from the comic strip "Bloom County," for all you youngsters.

Ack accepts two ASCII data files containing the waveforms to be compared. It assumes two columns of space-delimited, floating-point data; other formats will be regurgitated. The waveforms need not be on a common time axis with a common time increment. The time increment need not be constant from one data point to the next.

The first thing Ack does after reading the two files is to time shift one of them so that they both cross a user-specified threshold at the same point in time. Then it truncates the waveforms using the maximum leftmost time point and the minimum rightmost time point as the endpoints of the new interval. Before computing the figures of merit, Ack interpolates both waveforms onto a common time axis using a LaGrange polynomial interpolation function. If for some reason this interpolation routine breaks down – it is the most likely thing to break – I would recommend dropping the $100 or so and buying a copy of "Numerical Recipes in C." There's a routine called polint that works pretty well, but I could not use it because of copyrights.

The first figure of merit (FOM1) is simply the average of the absolute value of the relative errors, as defined in the "I/O Buffer Accuracy Handbook." This figure of merit weighs the differences between the two waveforms against a user-specified scaling factor, YW, usually the logic swing.

$$FOM1 = 100 \cdot \left[1 - \frac{\sum_{i=1}^{N} \left| Y1_i - Y2_i \right|}{YW \cdot N} \right]$$

where N is the number of data points, Y1 is the y coordinate of waveform 1, and Y2 is the y coordinate of waveform 2. If both waveforms are identical, this metric should yield 100%. However, it may produce misleading results in cases where the waveforms fit poorly in a small region but fit well in other regions, weighing the average in favor of the well-fitting regions. Therefore, we need two other metrics, the maximum relative error (FOM2) and the maximum absolute error (FOM3).

$$FOM2 = 100 \cdot \left[\frac{\max \left| Y1_i - Y2_i \right|}{YW} \right]$$

$$FOM3 = \max \left| Y1_i - Y2_i \right|$$

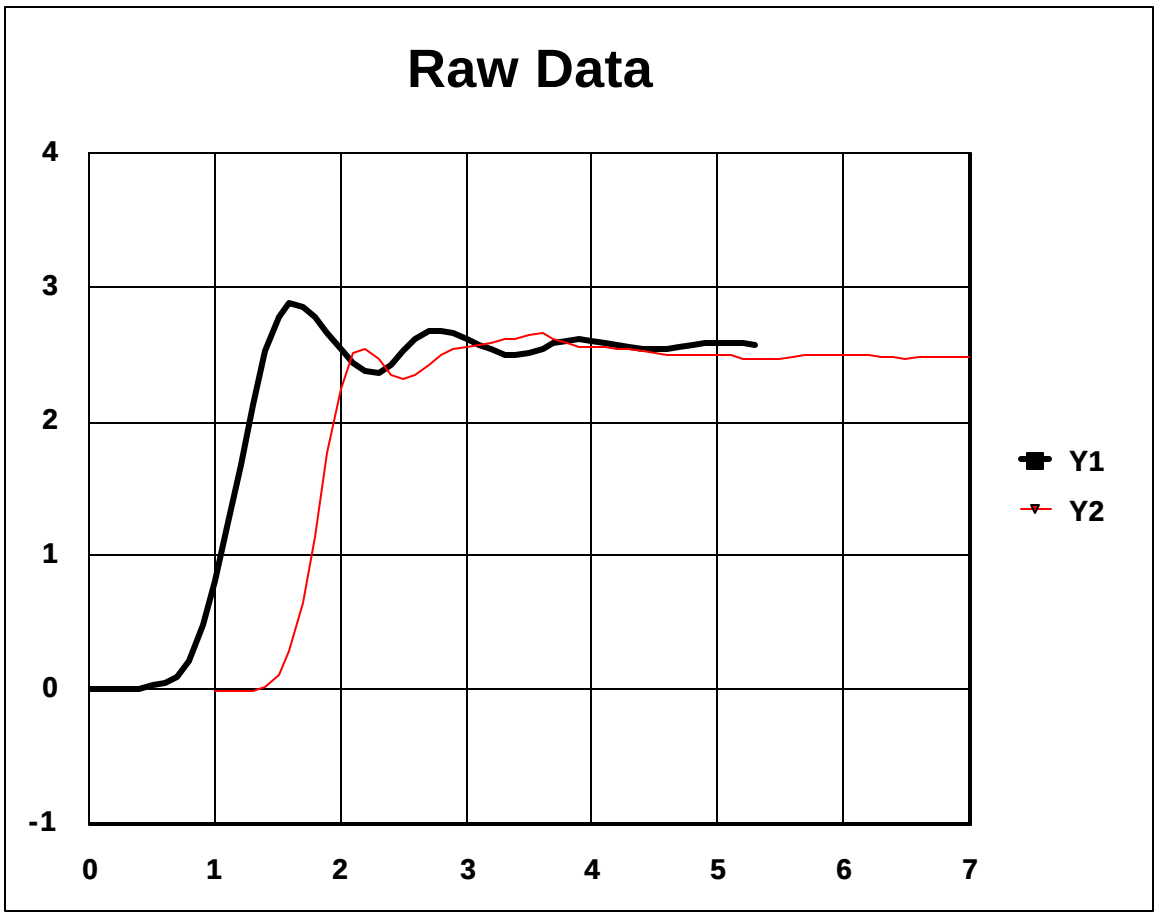
Ack dumps the three figures of merit to the standard error stream. It sends the two waveforms (three columns of floating-point data) to the standard I/O so that you may redirect them to a file.

I'm sure a good Perl programmer could have written this code in half the lines or less. It seems that I just got done making the transition from Pascal to C when someone decided that Perl was the next big thing. Pascal was a more idiot-proof language – and prettier, too. Judging by the number of core dumps I run into, I'd say that there are a lot of idiots out there that need proofing. (That's not to say this program is bug-free. It's certainly far from release-level code.) I'm sure Perl is much more convenient for parsing data streams. Perhaps we should mix Pascal and Perl and just chuck C. Anyhow, IBM's lawyers gave me their blessing to publish this code, so here it is.

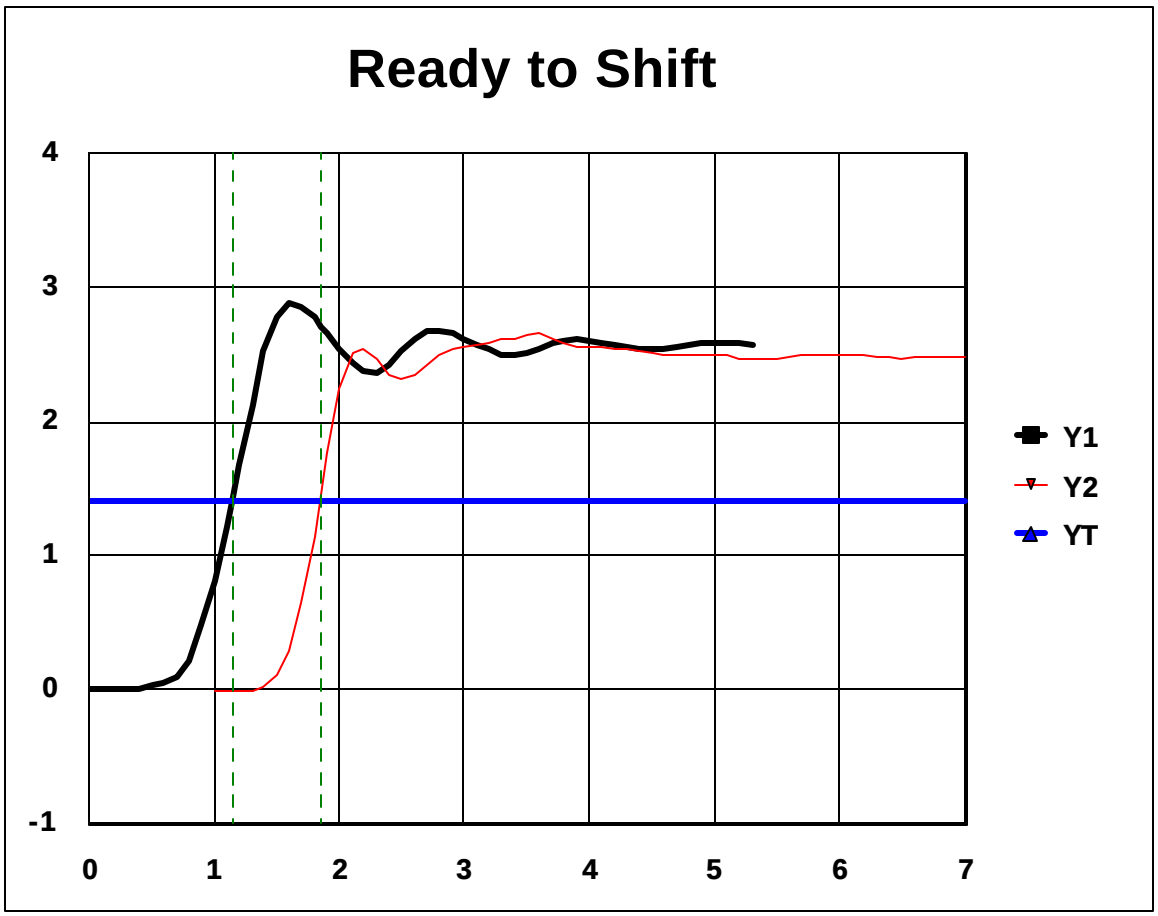
Usage: ack -f1 file1 -f2 file2 -e edge -dx dx -xu xu -yt yt -yw yw

Where:

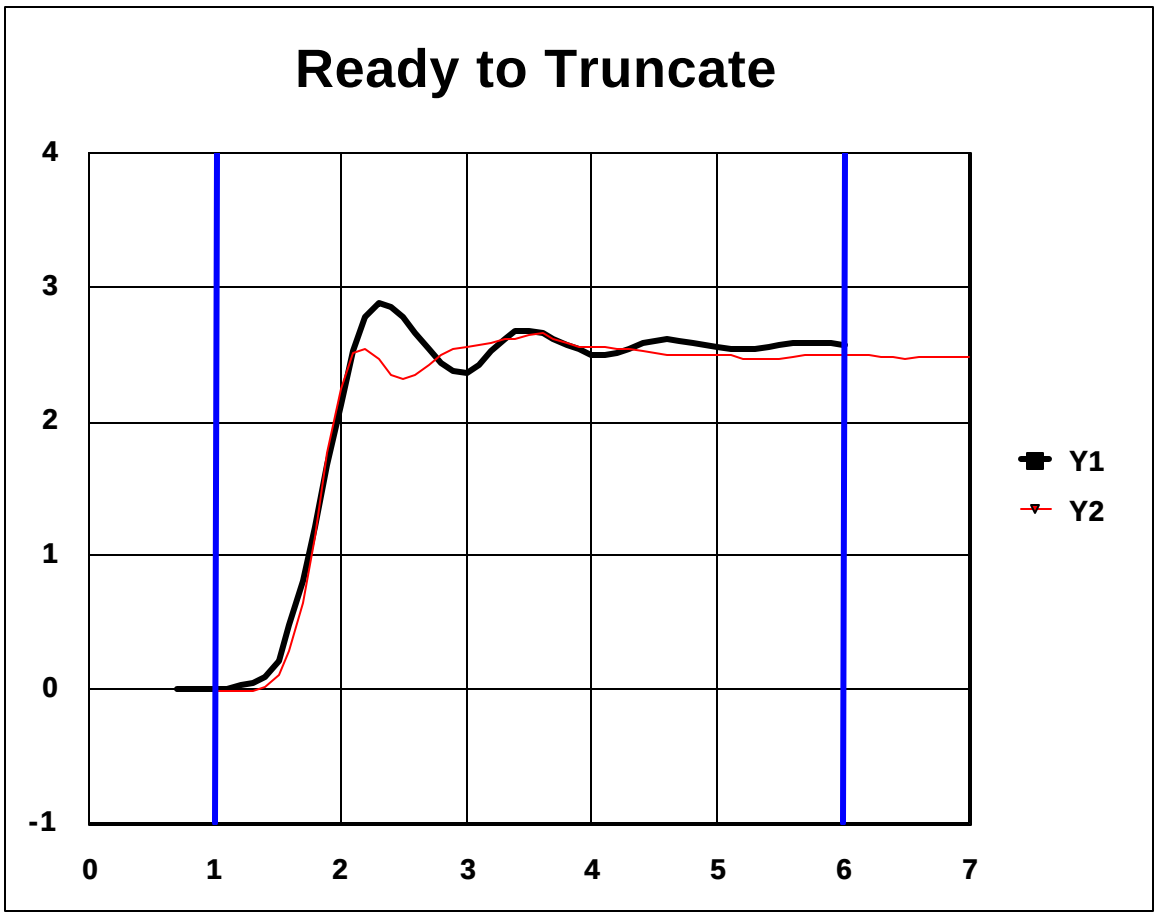
file1	= two columns of floating-point data, space-delimited. required.
file2	= two columns of floating-point data, space-delimited. required.
edge	= r for rising and f for falling. optional. default = r
dx	= desired x increment in units of xu. required.
xu	= units for scaling x input data. optional. default = 1e-9
yt	= y threshold used to shift waveform. optional. default = 0.5
yw	= weighing factor used to compute average. optional. default = 1.0



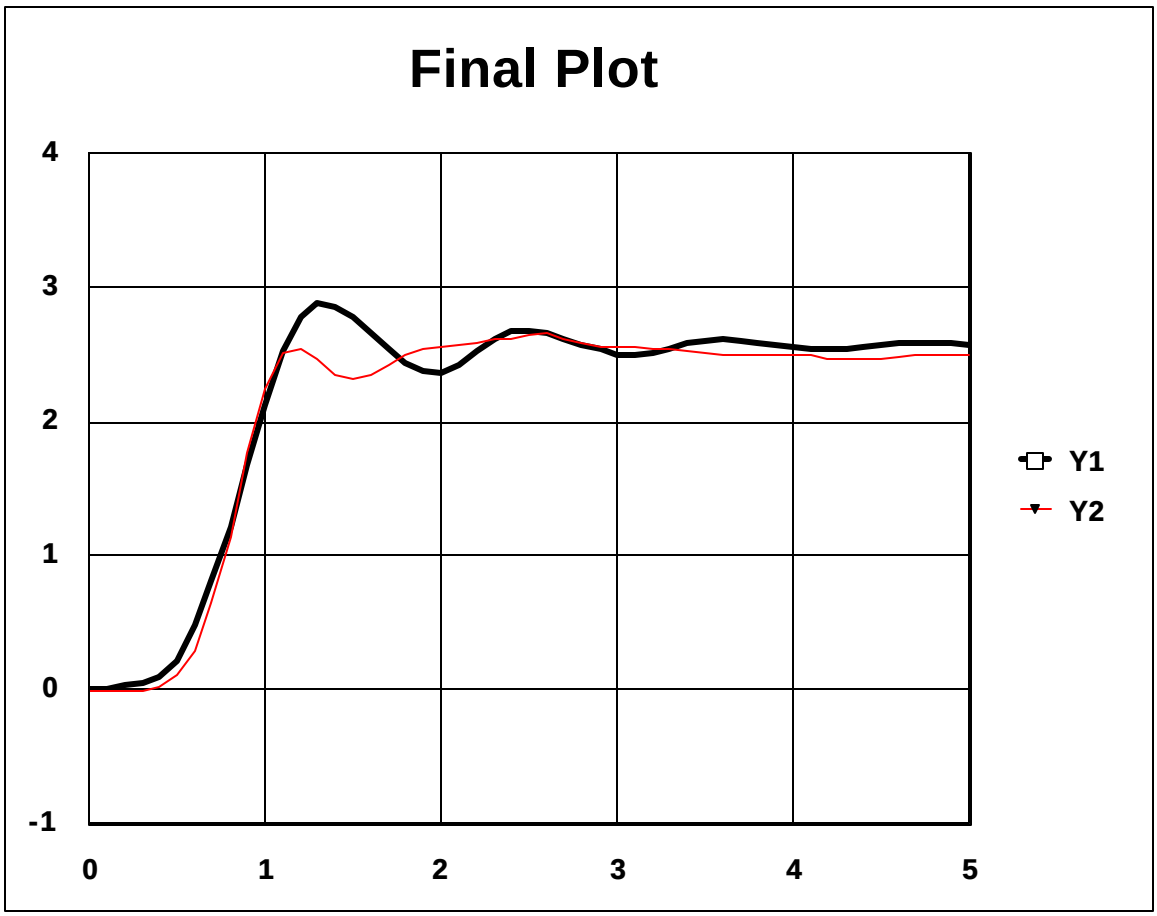
After parsing the command line options, Ack reads waveform 1 (variables x1, y1) from file1 and waveform 2 (variables x2, y2) from file2. It divides each x1 and x2 value by the command line option xu. If the input data are in units of seconds, I recommend using xu = 1e-9 to help the interpolation, which works better when there are not large differences in the orders of magnitude of numbers.



The next thing Ack does is to find the two x values (variables xx1 and xx2) where each waveform crosses the threshold specified by the yt command line option. It does this using simple linear interpolation (y = mx + b). In this step, it is important to tell Ack whether the first edge in the waveform is rising or falling using the edge command line option.



Now that Ack knows the where both waveforms cross the threshold, it is possible to shift one waveform so that it overlays the other. Ack does this by finding the leftmost waveform and adding a phase shift to each x value, thereby shifting it right. At this point, the original time increment of each waveform is still preserved. The bold vertical bars represent the truncation interval. After truncation, the program generates a new x-axis using the command line option starting at x = 0 and incrementing by the command line option dx. It then interpolates both waveforms onto the new x-axis using LaGrange polynomial interpolation.



The final waveforms are now on a common x-axis and can be exported to a three-column ASCII data file. The program computes the figures of merit and dumps them to the Unix standard error output.

References

I/O Buffer Information Specification 1.1, 2.1, 3.2
IBIS Cookbook 2.0
I/O Buffer Accuracy Handbook
IBIS Accuracy Test Board Design Data
IBIS Accuracy Test Board Application Note

The above documents are available on the IBIS web site: www.eia.org/eig/ibis/ibis.htm.

"Measuring Parasitic Capacitance and Inductance Using TDR," David J. Dascher, Hewlett-Packard Journal, April 1996, www.hp.com/hpj/apr96/ap96a11.htm.

Haller, R, and G Edlund, "Constructing Accurate Models of Behavioral I/O buffers," Designcon98 proceedings, ISSN 0886-229X, ISBN 0-933-217-47-1, 1998.