

Fundamental frequency -- limited resources

Source: <http://sci.tech-archive.net/Archive/sci.math.num-analysis/2004-08/0281.html>

From: Email Unread (*aa99aa_at_verizonmail.com*)

Date: 08/18/04

Date: 18 Aug 2004 04:50:57 -0700

Hi --

Hope this is being posted to the appropriate group(s) -- I didn't want to carpet-bomb all the NG's but I figured these 2 groups would be the most helpful.

After banging my head into a wall for many days on a particular problem, I realized I might be able to benefit from the wisdom of these newsgroups.

I have inherited a very resource-constrained embedded system, primarily performing data acquisition. Memory is the biggest constraint, the processor horsepower is not as big a constraint.

Anyways, suppose I have system sampling (via ADC) at 10,000 samples/sec. Suppose several times per second, I get an "event", which is a rectified voltage spike. For the sake of this example, let's suppose I get an event every 10ms, i.e. every 100 samples. Thus every 100th sample would be non-zero, the rest would be zero. (Let's assume no noise, each event is one sample in width, etc...)

So if I had an array of 50,000 elements, which would represent 5 seconds of data, with most of the samples reading "0", and about 500 elements reading non-zero when an event occurred. So far, so good....

Now here is the issue... there isn't memory to store 50k, 100k, etc... samples. This is a small 16 bit processor board designed years ago with limited RAM, not the common 32-bit processor running with 128M of RAM like so many "embedded" systems today.

Also, to make things "worse", sometimes an event is missed, and sometimes a "ghost event" shows up 1/2 way between real events.

So what I am saying is this... instead of an event every 100 samples, sometimes I will get an additional event at sample 150, 550, 750, etc... and sometimes we will get an event at 600 and 800, but not 700.

These "event glitches" are due to shortcomings in the system hardware and cannot be fixed.

The system is used to measure the linear speed of a moving system, the closer the events are to each other (i.e. smaller period), the faster the machinery is moving. The resource constrained system must frequently estimate the speed of the system by measuring the time between events.

In a perfect world, with no missed events and no false events, we could just measure the period between the last 2 events, and that would be the best (most recent) estimate of the system speed.

In an almost-perfect world (i.e. non-real time, post-processing on a PC, etc...), an FFT could be used on a huge array of time-domain samples, and the fundamental frequency of 100 would come through, in spite of the "ghost events" and missing events.

But this system has limited memory, and it interrupts every 500 samples or so, and the driver stores indices of non-zero events. In other words, over 1 second (10k samples), if events are happening every 100 samples, the driver would store the values 100, 200, 300, etc... 9900, 10,000 in a small array.

The reality is that the array will actually hold values like ...4200, 4250, 4300, 4400, 4600, 4650, 4700, etc.... due to false & duplicate events.

I have tried things like a histogram (find the "mode" of the period), a median filter, etc... but they are all susceptible to problems.... I keep thinking, "if I could just do an FFT and take the dominant freq, I'd be set!"

So the \$64,000 question is this is there some way to find the "fundamental frequency" of this data (which would be 100 in this case), assuming I don't have the memory to save (or re-create) a huge time-domain array of samples, and perform an FFT?

(Humility note: When I took this little project over, I thought, "no problem, visually, I can easily what the period is, this will be a piece of cake in firmware....")