COMPARISONS OF THE EXECUTION TIMES AND MEMORY REQUIREMENTS FOR HIGH-SPEED DISCRETE FOURIER TRANSFORMS AND FAST FOURIER TRANSFORMS, FOR THE MEASUREMENT OF AC POWER HARMONICS

A. J. Roscoe [1], *G. M. Burt*[1]
[1] University of Strathclyde, Glasgow, UK
E-mail (corresponding author): andrew.roscoe@eee.strath.ac.uk

Abstract – Conventional wisdom dictates that a Fast Fourier Transform (FFT) will be a more computationally effective method for measuring multiple harmonics than a Discrete Fourier Transform (DFT) approach. However, in this paper it is shown that carefully coded discrete transforms which distribute their computational load over many frames can be made to produce results in shorter execution times than the FFT approach, even for large number of harmonic measurement frequencies. This is because the execution time of the presented DFT actually rises with N and not the classical N^2 value, while the execution time of the FFT rises with $N\log_2 N$.

Keywords Power system harmonics, Harmonic analysis, Fourier transforms, Power quality.

1. INTRODUCTION

Traditionally, accurate measurement of voltage or current harmonics within AC power systems can be made over relatively long timeframes, with relatively low update rates. For example, [1] specifies that "class A" instruments measuring power quality shall do so over 10 cycles (for 50 Hz systems) or 12 cycles (for 60 Hz systems), with further aggregation stages to provide 150/180-cycle and 10-minute averages. Such pieces of equipment allow standards such as [2] to be assessed, which specify power system performance over such 10-minute intervals.

However, new requirements for metering, real-time power quality assessment, inverter control, and active control of harmonic contamination, all require accurate measurement of harmonic content at much higher update rates. For example, the IEEE specification for PMU (Phase Measurement Unit) performance C37.118-2005 [3] specifies update rates of 0,1 Hz to 25 or 30 Hz (2 cycles, for 50 and 60 Hz systems, respectively). A power-electronic device actively mitigating harmonic contamination might require an update at its switching frequency. To accurately assess harmonic content including both even and odd harmonics, making the measurements over an exact number of cycles is highly desirable since it minimises the spectral leakage of any Fourier transform applied to the data, which maximises the accuracy of the results and minimises the real-time ripple on the results. Failing to correctly implement such algorithms can result in poor accuracy and ripple for off-nominal frequencies [4] [5].

It should be noted that the update rate can be much higher than the fundamental frequency, even though the measurements consider full cycle(s) of data. This is possible if the measurement algorithms consider the entire dataset every computational frame, coherent with the ADC (analogue to digital converter) sample rate, and output a new result every frame or every few frames.

In this paper, two distinct methods are presented which are able to make such measurements over exactly 1 cycle. Both methods assume that the measuring equipment sample rate is fixed. This differs from some existing types of measuring equipment which modify their sample rates to match the AC power frequency. The first method involves carefully and quickly re-sampling the sampled waveform in such a way that exactly 2^n samples fall within one fundamental period, when n is integer. A standard FFT (Fast Fourier Transform) can then be used to reveal the harmonic analysis, with zero or minimal spectral leakage [6] [7]. The second method uses Discrete Fourier Transforms (DFTs) to measure each and every harmonic of interest. While intuitively this will provide a more inefficient solution, the DFTs are implemented using carefully coded rolling buffers and integrators, which minimises the numerical calculations per frame [8] [9]. This leads to some counter-intuitive results which are presented later.

In the following sections, these two algorithms are described in greater detail. Then, the two algorithms are tested in real-time using two candidate processors, to assess the actual achievable execution times and required data memory.

2. MEASUREMENT METHODS

Both methods presented assume that the incoming data is sampled at a suitable data rate, which in this paper is assumed to be at twice the Nyquist frequency of the highest harmonic to be measured (Oversampling factor m_O=2) at nominal frequency, or at a sample frequency high enough to ensure aliasing does not corrupt the measurements. For example:

$$T_s = \frac{1}{2 m_o f_0 H_{\max}} . \quad (1)$$

where T_s is the sample time and computational frame time (reciprocal of sample frequency and frame rate), m_0 is the oversampling factor, f_0 is the nominal frequency, and H_{max} is the highest order harmonic to be measured (or required to avoid aliasing).

The samples from the ADC (analogue to digital converter) flow with a fixed time interval of T_s in both methods presented. It is assumed that a measurement of the fundamental frequency is available. Indeed, frequency can be measured by $d\Phi/dt$ of the fundamental using the methods described in this paper or [8].

Both methods presented in this paper measure the fundamental and harmonics over 1 exact cycle period. This limits the analysis to exact harmonics, and precludes accurate analysis of inter-harmonics. Both methods could be adjusted to do this, but would require measurement over longer integer numbers of cycles to enhance the frequency resolution while still minimising spectral leakage [6].

The algorithms in this paper are coded in MATLAB® Simulink code, and then compiled into 'C' code for target processors using the Real-Time Workshop and Embedded Coder toolboxes. This provides platform independence and a robust development environment. The major benchmarking activities have been carried out on the 32-bit Infineon TC1796 microcontroller [10], which is targeted at automotive applications but is also highly suitable for power-electronic applications. For reference, in this paper, the programs were executed from internal flash memory via the CPU cache at 0x80000000, using the internal 56kB and 64kB RAM sections at 0xD0000000 and 0xC0000000.

2.1. FFT measurement method

For the FFT method, the challenge is to re-sample the data into a new data stream with a different sample rate, such that 2^n samples cover exactly the period of the fundamental signal, where 2^n is selected such that it is large enough to provide at least the same level of oversampling m_O as provided by the ADC sample rate, at the nominal frequency f_0. This method is described in [6], and an overview is shown in Fig. 1. However, in this paper significant effort has been taken to optimise the implementation.

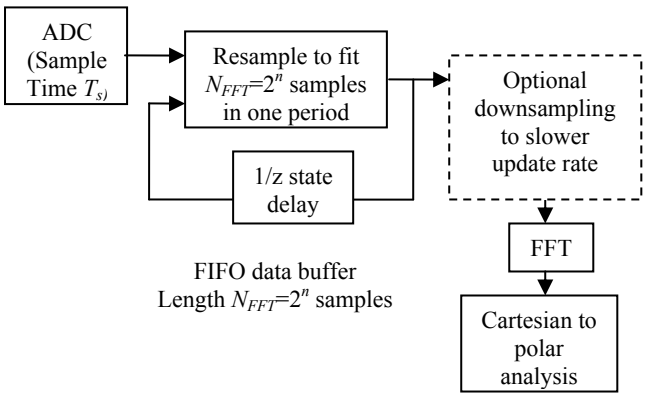

Fig. 1. Overview of FFT algorithm.

Firstly, the third-order interpolation using the Newton Interpolation Formula is optimised relative to [6]. The algorithm is shown in Fig. 2, which provides interpolation backwards in time by fractional proportions of the ADC sample time T_s.

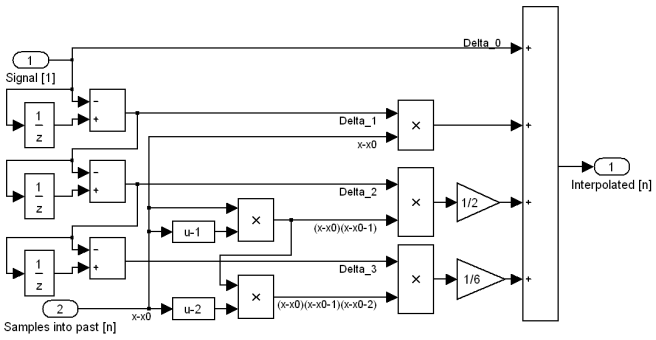

Fig. 2. Representation of 3rd order interpolation algorithm

However, the actual algorithm cannot usefully be coded directly in Simulink, since it requires asynchronous sample rate conversion, from the fixed sample time T_s to the varying sample time required to fill 2^n samples in exactly one fundamental period. In this case, the simplest solution is to write some Embedded MATLAB script which carries out the task. This continuously updates a rolling buffer of 2^n samples, bringing in one or more new interpolated samples each frame, and over-writing the oldest ones in a FIFO (first-in first-out) nature. However, when built, this is results in 'memcpy' operations in the 'C' code, which wastes precious CPU time. Therefore, the algorithm has been coded in a 'fully in-lined' Simulink 'S function', using 'Work vectors' to manage the buffer of 2^n samples. The most informative parts of this algorithm are shown in Fig. 3, which is an excerpt from the 'S function' 'C' code. In this algorithm, `Tlag_decrement` is the reciprocal of the number of new FFT samples which need to be generated each computational frame, i.e. the ratio of the required FFT sample rate to the fixed ADC sample rate.

```
    int32_T  *FFT_ptr = ssGetIWork(S);
    real_T   *xD = ssGetRWork(S);
     int_T    ok = 1;
     real_T   t,a,D1,D2,D3;

     D1 = xD[0] - (*Signal);
     D2 = xD[1] - D1;
     D3 = xD[2] - D2;
     *Tlag = xD[3] + 1;
     while (ok) {
         t = (*Tlag) - (*Tlag_decrement);
         ok = (t>=0);
         if (ok) {
             (*Tlag) = t;
             /* 1st order interpolation */
             a = (*Signal) + D1*(*Tlag);
             t = (*Tlag) * ((*Tlag)-1);
             a += D2*t*0.5;
             /* 2nd order interpolation */
             t *= ((*Tlag)-2);
             a += D3*t/6;
             /* 3rd order interpolation */
             /* bump the FIFO buffer */
             (*FFT_ptr)++;
             if ( (*FFT_ptr) > (N_FFT-1) ) {
                 *FFT_ptr=0;
             }
             FFT_Data[*FFT_ptr] = a;
         }
     }
     *FFT_pointer = *FFT_ptr;
 /* Updates for states */
     xD[0] = *Signal;
     xD[1] = D1;
     xD[2] = D2;
     xD[3] = *Tlag;
```

Fig. 3. Extract from the Simulink 'S function' for re-sampling from T_s to the FFT FIFO buffer.

By fully in-lining the 'S function', the execution time of the re-sampling is reduced to less than 0.9µs per computational frame.

Next, the 2^n samples pass to an FFT. The Simulink FFT block is used, which is well optimised and automatically recognises that the input data is real (not complex) and reduces the 2^n sample FFT to a $2^{(n-1)}$ FFT [11]. Finally, the required fundamental and harmonic measurements are extracted from the FFT output. The Cartesian to polar analysis requires the use of sqrt() and atan2() functions which are computationally expensive [9]. In addition, when referencing the harmonic phases to the fundamental phase, care is taken to avoid the use of the Simulink 'MOD' function to keep phases within the range of $-\pi$ to $+\pi$ since this can take up to 2.3µs per operation [9]. Instead, native casting from floating-point to integer types in C is used to create a manually coded 'MOD' function, taking care to account for the variant behaviours of different target processors [9]. This drops the execution time for 'MOD' to less than 0.4µs per operation. Even so, the amplitude and phase analysis of 40 harmonics, can be a significant proportion of the entire algorithm execution time, as shown later.

While the re-sampling is very fast when implemented in the FIFO fashion on a continuous basis, both the FFT operation and the magnitude/phase analysis can be time consuming. In particular, the FFT operation has to analyse the entire dataset each time it is executed. In the FFT algorithm, the option exists to only carry out the FFT operation and final analysis at a much lower data rate than that of the sampled ADC data, potentially using a low-priority background task. The only part of the algorithm which must be executed with high priority at the sample time T_s is the re-sampling and maintenance of the FIFO buffer integrity.

In terms of data memory use, the FFT algorithm is very efficient. For an algorithm using an $N_{FFT}=2^n$ point FFT, the dominant data memory required (assuming 32-bit arithmetic) is $4*N_{FFT}$ bytes for the FIFO buffer, $8*N_{FFT}$ bytes for the FFT (at its output, although it is evaluated as a $2^{(n-1)}$ point FFT), and $4*N_{FFT}*0.75$ for a "twiddle" array used inside the Simulink FFT algorithm.

2.2. DFT measurement method

The DFT method builds simply upon the method described in [8], using the optimisations described in [9] which minimise the execution time. A high-level view of the DFT algorithm is shown in Fig. 4. In this method, every harmonic to be analysed is subjected to a DFT analysis, by correlation with sin() and cos() waveforms at the appropriate harmonic frequencies (Fig. 5), and evaluation of the definite integrals of the correlations over exactly one fundamental period (Fig. 6).

This is achieved by continuously integrating the correlations and storing the results in rolling buffers, each of which must be long enough to store a full period of the lowest frequency f_{min} which can be analysed accurately. Typically this can be set to about $f_{min}=0.8*f_0$ (nominal) for most power system operations, but can be set lower for specialised applications (at the expense of additional memory requirement).

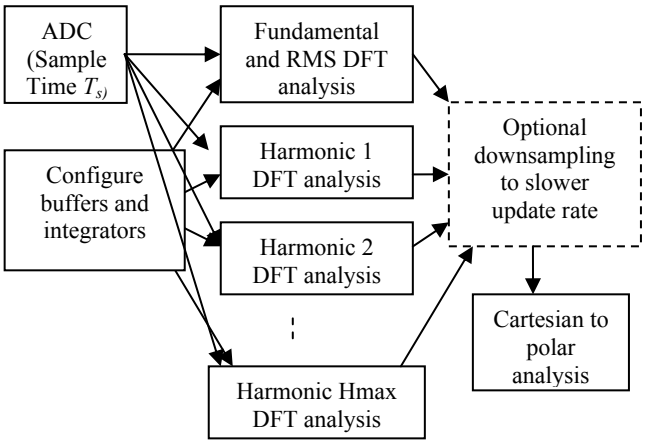

Fig. 4. Overview of DFT algorithm.

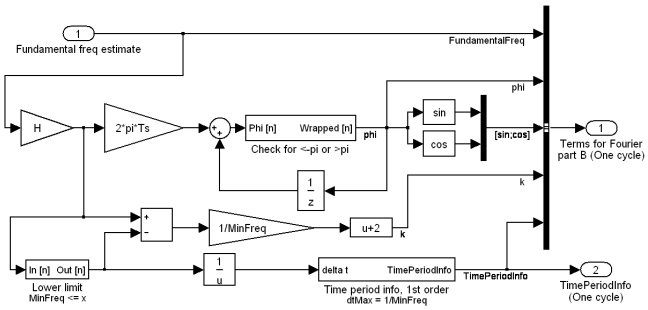

Fig. 5. Configuration of correlations for a single harmonic, and configuration of buffers (the `TimePeriodInfo` signal) which is common for all harmonics

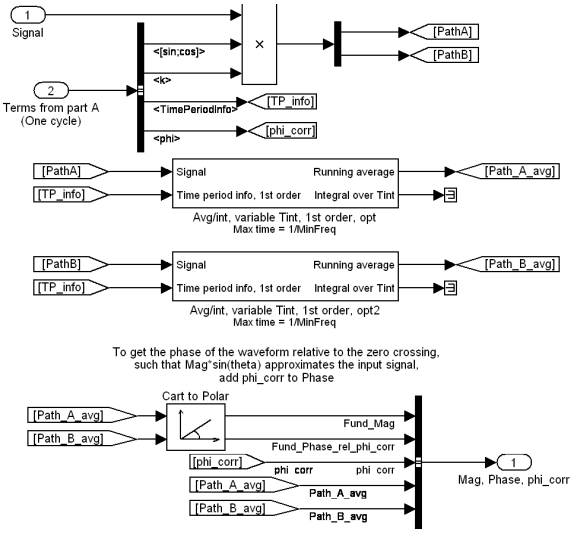

Fig. 6. Evaluation of DFT correlations for the fundamental or a single harmonic

The definite integrals are evaluated by subtracting the integrator output at a previous time, exactly one fundamental period in the past, from the most recent integrator output. The complications are that this time is generally not an integer multiple of the sample time T_s, and

that the integrator can tend to wind up. For this reason, not one but 3 buffers are required to evaluate each integral: two to form a pair of integrators operating in a tick-tock scheme, and a third to carry out the 1st-order linear interpolation to account for the 'part sample' effect (Fig. 7, Fig. 8, Fig. 9 and [8]). The tick-tock pair are operated with each integrator reset to zero once every few cycles, then left to acquire at least one full cycle of data and become valid, and then used until the other integrator path becomes valid. A 2-buffer variant is possible [8], but introduces a varying latency which may be undesirable in active control applications.

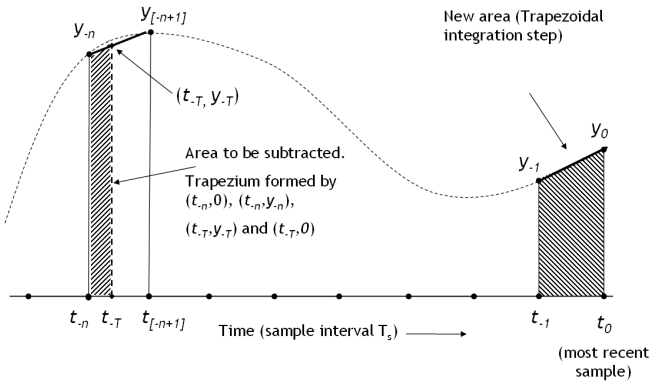

Fig. 7. The procedure for performing exact-time averaging [8].

Since the analysis of every harmonic occurs over the same single-cycle period, every buffer is configured the same way, and this configuration, including a large part of the 1st-order interpolation calculations, only need to be carried out once for the entire set of harmonics, each frame, based upon the estimation of fundamental frequency. Thus, in Fig. 5, the block which generates the `TimePeriodInfo` signal only needs to be executed for the fundamental. The analyses for the higher harmonics re-use the same information.

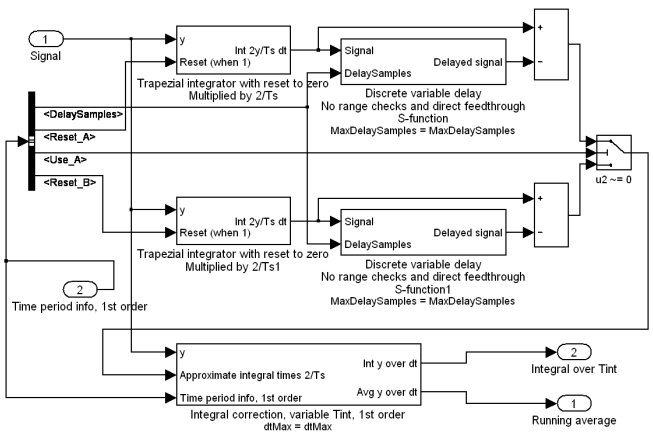

Fig. 8. Exact-time averaging code, showing twin integrators in tick-tock configuration, and the interpolation block.

The output of the definite integrators form the complex values of the fundamental and harmonic components, which can then be related together and converted to amplitude/phase in a similar way to the FFT analysis.

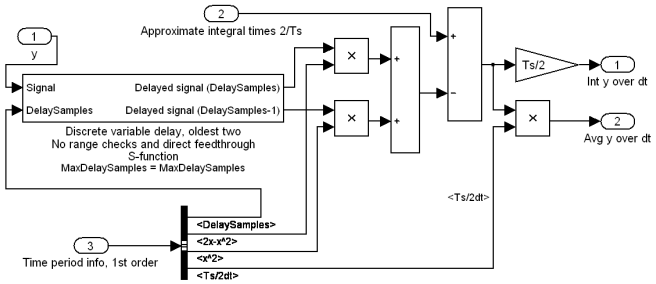

Fig. 9. Correction of the integral to interpolate between the oldest samples so that the integral is over exactly one fundamental period, using pre-calculated buffer and interpolation parameters which are shared with all harmonic analyses.

By comparison with the FFT algorithm, the option exists to down-sample the final data before the final Cartesian to polar analysis, but apart from that, the entire algorithm must be executed at the sample time T_s. That having been said, while the FFT operation needs to examine the entire dataset every time it is executed, the beauty of the DFT algorithm using the rolling buffers is that only a tiny part of the Fourier Transform has to be calculated each time a new ADC sample arrives. Essentially, the DFT computation is spread evenly over a single fundamental period, and is continuously updated.

In terms of data memory use, the DFT algorithm is relatively heavy. The requirement is 9 buffers for the fundamental (3 each for each sin() and cos() integral, and 3 more can be used to allow evaluation of the overall RMS (Root-Mean-Square) and THD (Total Harmonic Distortion) figures), plus 6 buffers for each harmonic to be measured. The length of the buffers is $(1/f_{min}/T_s+2)$, requiring 4 times this amount of bytes assuming 32-bit arithmetic is used.

3. BENCHMARKING RESULTS

The algorithms were initially benchmarked on the Infineon TC1796 microcontroller, in a similar manner to that described in [9]. Care was taken to incrementally add reference code (test signal generators) and then the main pieces of algorithm code, so that true execution times were measured. The times were measured using a toggled output line to indicate the beginning and end of the test algorithm, and a digital scope to measure the pulse widths. Multiple repetitions were used to increase the measurement accuracy, particularly for smaller algorithmic sections with short execution times.

The first set of results (Fig. 10) show the execution times of the two methods (FFT and DFT) when required to measure the fundamental and harmonics up to (and including) a value H_{max} which was varied between 1 to 40. In this analysis, the ADC sample time T_s varies with the required maximum harmonic by (1). Over-sampling m_0 is set at 2. It is assumed that harmonics above the required measured set are attenuated in analogue filters to avoid aliasing effects.

Some of the key parameters of the two algorithms during this test are shown in Table I, for a nominal value of f_0=50 Hz.

TABLE I. Parameters for flexible ADC sample-rate test

H_{max} = Harmonics To analyse	$1/T_s$	DFT buffer length (floats)	N_{FFT}	FFT Sample rate (for $f=f_0$)
1	200 Hz	7	8	400 Hz
5	1 kHz	27	32	1,6 kHz
11	2,2 kHz	57	64	3,2 kHz
21	4,2 kHz	107	128	6,4 kHz
31	6,2 kHz	157	128	6,4 kHz
40	8 kHz	202	256	12,8 kHz

Fig. 10 shows the resulting execution times, which are also broken down for the FFT algorithm to show the times required for the actual FFT operation, and the cartesian to polar analysis. The re-sampling takes less than 0.9µs per frame. Two lines are shown for the DFT algorithms. These are optimistic and pessimistic values for the TC1796, and the variation occurs depending upon the RAM (random access memory) speed. When larger quantities of memory are being accessed quickly, it can take longer for each access due (presumably) to the lowered ability of the CPU to cache the active memory segments. The red dashed line shows the limit at which the algorithms cannot be executed on the TC1796 within the allowed frame time T_s without down-sampling at least part of the analysis.

Fig. 10 shows that there is little to choose in execution time between the two methods.

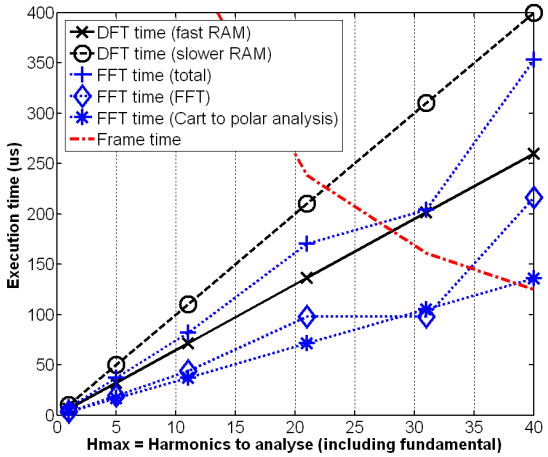

Fig. 10. Execution times on the TC1796. ADC sample rate set for 2x over-sampling at the highest harmonic to analyse.

Fig. 11 shows the data memory requirements of the two algorithms. Clearly, the DFT algorithm requires much more memory, which in this test rises as H_{max}^2 due to both the number of buffers, and the buffer lengths, rising with H_{max}. The required data memory for the FFT algorithm rises only with H_{max}. For the TC1796 processor, the maximum contiguous RAM segment with fast access speed is 64kB, which constrained the actual DFT benchmarking experiments to H_{max}<=21. The DFT results for H_{max}>21 in Fig. 10 and Fig. 11 have been carefully calculated and extrapolated, as if more contiguous memory was genuinely available.

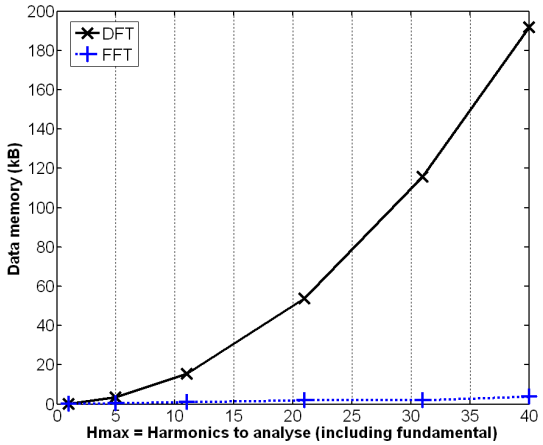

Fig. 11. Data memory requirement. ADC sample rate set for 2x oversampling at the highest harmonic to analyse.

Next, a similar test assumes that the ADC sample rate must remain fixed at 8 kHz to avoid aliasing, but that H_{max} varies as before. In this case N_{FFT}, the DFT buffer length, and the FFT sample rate, are all fixed at their values in the bottom row of Table I. The resulting execution times are shown in Fig. 12. The DFT algorithm is clearly faster when only the low orders of harmonics need to be measured directly.

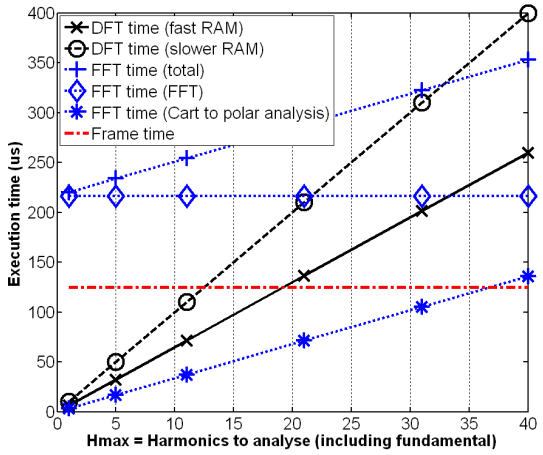

Fig. 12. Execution times on the TC1796. ADC sample rate 8kHz.

Finally, the analysis using variable ADC sample time (Table I) is repeated using the MVME5500 PowerPC card [12] using the MPC7457 processor [13], embedded with a VME rack system [14]. This card has 512MB of memory and a 512kB on-chip cache, and is easily capable of handling the data memory requirement of even the DFT analysis to the 40th harmonic and way beyond. The data memory requirement is doubled compared to Fig. 11, only because 64-bit arithmetic is applied by default by MATLAB for this target. The execution times (Fig. 13) are roughly 40% of the TC1796 times, and the DFT analysis is shown to be more clearly favourable over the FFT analysis than in Fig. 10, probably due to the faster memory access of the MVME5500 and its ability to quickly access all the rolling buffers every frame.

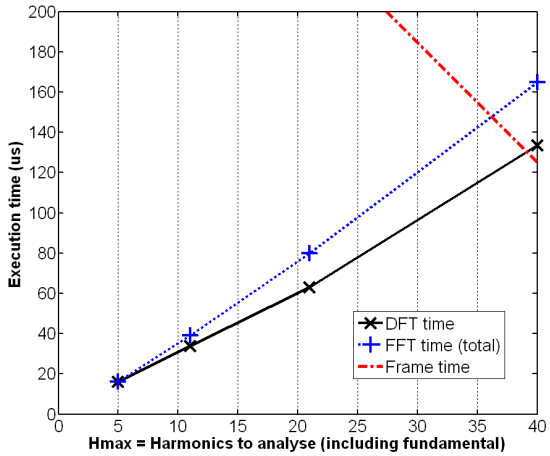

Fig. 13. Execution times on the MVME5500. ADC sample rate set for 2x oversampling at the highest harmonic to analyse.

4. CONCLUSION

While the FFT (Fast Fourier Transform) is generally regarded as the faster way to analyse waveforms than the DFT (Discrete Fourier Transform), it is found in this paper that in the application of electrical power systems, this is not always the case. Whereas intuition might lead to the suspicion that the DFT might be faster than the FFT at analysing small numbers of harmonics, but slower for analysing larger numbers of harmonics, in fact the DFT method can be competitive or faster than the FFT method for all numbers of harmonics.

The time taken for the core of the FFT algorithm to be performed rises as N_{FFT}*$\log_2(N_{FFT})$ [11] where N_{FFT} rises with the highest harmonic H_{max} which needs to be analysed. In contrast, while the execution time of a classical DFT would rise with N*H_{max}, where N is the number of DFT time points, in the presented algorithm the DFT only needs to perform part of the analysis every frame, and the analysis is spread out over many frames spanning one fundamental period. As a result, the execution time for the DFT algorithm only rises proportionately to H_{max}. Therefore, the DFT algorithm actually gets faster and faster compared to the FFT algorithm as H_{max} increases, by a factor of $\log_2(H_{max})$.

However, the memory requirement of the DFT algorithm is relatively large. While this is not an issue for some processors, for smaller microcontrollers the available memory may place hard limits on the number of harmonics which can be analysed, or the speed of the access to the wide memory segments may increase the execution time in a non-linear fashion.

Both of these algorithms evaluate sampled data over exactly one fundamental cycle. Both of these algorithms, but particularly the DFT algorithm, are suitable for creating data outputs at update rates much higher than once per cycle, potentially at the full ADC sample time T_s, due to the algorithm structure. This produces fast-responding measurements which can be used for metering or active harmonic control applications.

REFERENCES

[1] BSI, "Electromagnetic compatability (EMC) - Part 4-30:Testing and measurement techniques — Power quality measurement methods", BS EN 61000-4-3:2003, 2003.

[2] British Standards, "Voltage characteristics of electricity supplied by public distribution systems", 2000.

[3] K. E. Martin, D. Hamai, M. G. Adamiak, S. Anderson, M. Begovic, G. Benmouyal, G. Brunello, J. Burger, J. Y. Cai, B. Dickerson, V. Gharpure, B. Kennedy, D. Karlsson, A. G. Phadke, J. Salj, V. Skendzic, J. Sperr, Y. Song, C. Huntley, B. Kasztenny, and E. Price, "Exploring the IEEE standard C37.118-2005 synchrophasors for power systems", *IEEE Transactions on Power Delivery,* vol. 23, pp. 1805-1811, Oct 2008.

[4] Y. Hu and D. Novosel, "Progresses in PMU testing and calibration", *2008 Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, Vols 1-6,* pp. 150-155, 2008.

[5] J. Depablos, V. Centeno, A. G. Phadke, and M. Ingram, "Comparative testing of synchronized phasor measurement units", *2004 IEEE Power Engineering Society General Meeting, Vols 1 and 2,* pp. 948-954, 2004.

[6] G. W. Chang, C. I. Chen, Y. J. Liu, and M. C. Wu, "Measuring power system harmonics and interharmonics by an improved fast Fourier transform-based algorithm", *IET Generation Transmission & Distribution,* vol. 2, pp. 192-201, Mar 2008.

[7] H. Qian, R. X. Zhao, and T. Chen, "Interharmonics analysis based on interpolating windowed FFT algorithm", *IEEE Transactions on Power Delivery,* vol. 22, pp. 1064-1069, Apr 2007.

[8] A. J. Roscoe, G. M. Burt, and J. R. McDonald, "Frequency and fundamental signal measurement algorithms for distributed control and protection applications", *IET Generation, Transmission & Distribution* vol. 3, pp. 485-495, May 2009.

[9] A. J. Roscoe, S. M. Blair, and G. M. Burt, "Benchmarking and optimisation of Simulink code using Real-Time Workshop and Embedded Coder for inverter and microgrid control applications", in *Universities' Power Engineering Conference (UPEC)*, Glasgow, UK, 2009.

[10] Infineon Technologies, "Infineon Tricore family (TC1796)", Available: http://www.infineon.com, accessed January 2011.

[11] W. H. Press, S. A. Teukolski, W. T. Vetterling, and B. P. Flannery, "Numerical Recipes in C", Second ed, ISBN 0521 431085, 1992.

[12] Emerson Network Power, "MVME5500 VME with PowerPC Processor", Available: http://www.emersonnetworkpower.com, accessed January 2011.

[13] Freescale, "MPC7455: Host Processor", Available: http://www.freescale.com, accessed January 2011.

[14] Applied Dynamics International, "Real Time Station (RTS)", Available: http://www.adi.com, accessed January 2011.

Author (s):
Dr Andrew J. Roscoe, University of Strathclyde, Department of Electronic & Electrical Engineering, Royal College, 204 George Street, Glasgow, UK. andrew.roscoe@eee.strath.ac.uk

Prof Graeme M. Burt, University of Strathclyde, Department of Electronic & Electrical Engineering, Royal College, 204 George Street, Glasgow, UK. g.burt@eee.strath.ac.uk