Quantum Wave ToolBox

Documentation of Algorithms

QWTB version 0.1

https://qwtb.github.io/qwtb/

Contents

1

Introduction

This document gives overview of the algorithms implemented in Quantum Wave ToolBox (QWTB).

Toolbox was realized within the EMRP-Project SIB59 Q-Wave. The EMRP is jointly funded by the EMRP par- ticipating countries within EURAMET and the European Union.

2

INL – Integral Non-Linearity of ADC

Description

.id — INL

.name — Integral Non-Linearity of ADC

.desc — Calculates Integral Non-Linearity of an ADC. ADC has to sample sinewave, ADC codes are required.

.citation — Virosztek, T., Pálfi V., Renczes B., Kollár I., Balogh L., Sárhegyi A., Márkus J., Bilau Z. T., ADCTest project site: `http://www.mit.bme.hu/projects/adctest` 2000-2014

.remarks — Based on the ADCTest Toolbox v4.3, November 25, 2014.

.license — UNKNOWN

.requires

- t — time series of sampled data
- codes — Sampled values represented as ADC codes (not converted to voltage)

.returns

- INL — INL

.providesGUF — no

.providesMCM — no

Example

Integral Non Linearity of ADC

Example for algorithm INL

INL is an algorithm for estimating Integral Non-Linearity of an ADC. ADC has to sample sinewave, ADC codes are required.

See also 'Virosztek, T., Pálfi V., Renczes B., Kollár I., Balogh L., Sárhegyi A., Márkus J., Bilau Z. T., ADCTest project site: `http://www.mit.bme.hu/projects/adctest` 2000-2014';

Contents

Generate sample data

Suppose a sine wave of nominal frequency 10 Hz and nominal amplitude 1 V is sampled by ADC with bit resolution of 4. First quantities `t` with time of samples and quantity `bits` with number of bits are prepared and put into input data structure `DI`.

```
DI = [];
DI.t.v=[0:1/1e4:1-1/1e4];
DI.bits.v = 4;
```

Waveform is constructed.

```
Anom = 1; fnom = 2; phnom = 0;
wvfrm = Anom*sin(2*pi*fnom*DI.t.v + phnom);
```

Next code values are calculated. It is simulated by quantization and scaling of the sampled waveform. In real measurement code values can be obtained directly from the ADC. Suppose ADC range is -1..1.

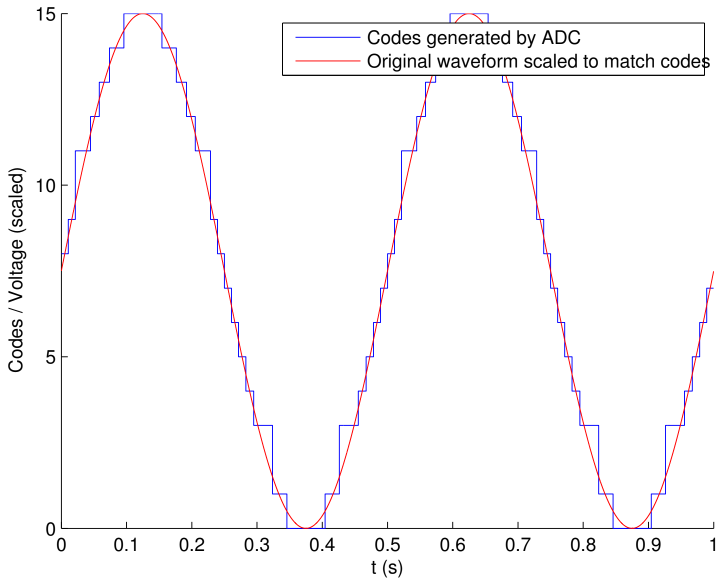
```
codes = wvfrm;
rmin = -1; rmax = 1;
levels = 2.^DI.bits.v - 1;
codes(codes<rmin) = rmin;
codes(codes>rmax) = rmax;
codes = round((codes-rmin)./2.*levels);
```

Now lets introduce ADC error. Instead of generating code 2 ADC erroneously generates code 3 and instead of 10 it generates 11.

```
codes(codes==2) = 3;
codes(codes==10) = 11;
codes = codes + min(codes);
```

Create quantity codes and plot a figure with sampled sine wave and codes.

```
DI.codes.v = codes;
figure
%plot(t, (y+1).*1/2.*levels, t, codes);
hold on
stairs(DI.t.v, codes);
wvfrm = (wvfrm + Anom).*levels./2;
plot(DI.t.v, wvfrm, '-r');
xlabel('t (s)')
ylabel('Codes / Voltage (scaled)');
legend('Codes generated by ADC','Original waveform
   scaled to match codes');
hold off
```



Call algorithm

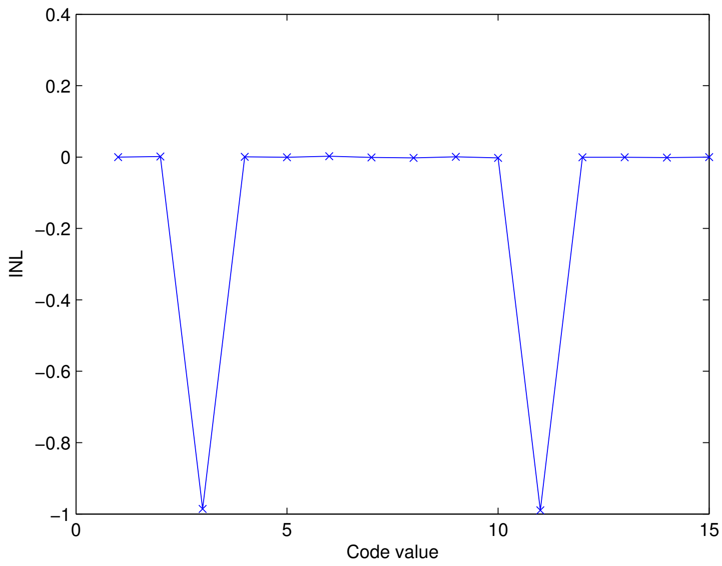
Apply INL algorithm to the input data DI.

```
DO = qwtb('INL', DI);
```

```
QWTB: no uncertainty calculation
```

Plot results of integral non-linearity. One can clearly observe defects on codes 3 and 11.

```
figure
plot(DO.INL.v, '-x');
xlabel('Code value')
ylabel('INL')
```


INL
Code value
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
5
10
15

3

PSFE – Phase Sensitive Frequency Estimator

Description

.id — PSFE

.name — Phase Sensitive Frequency Estimator

.desc — An algorithm for estimating the frequency, amplitude, and phase of the fundamental component in harmonically distorted waveforms. The algorithm minimizes the phase difference between the sine model and the sampled waveform by effectively minimizing the influence of the harmonic components. It uses a three-parameter sine-fitting algorithm for all phase calculations. The resulting estimates show up to two orders of magnitude smaller sensitivity to harmonic distortions than the results of the four-parameter sine fitting algorithm.

.citation — Lapuh, R., "Estimating the Fundamental Component of Harmonically Distorted Signals From Noncoherently Sampled Data," Instrumentation and Measurement, IEEE Transactions on , vol.64, no.6, pp.1419,1424, June 2015, doi: 10.1109/TIM.2015.2401211, URL: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7061456&isnumber=7104190`

.remarks — Very small errors, effective for harmonically distorted signals.

.license — UNKNOWN

.requires

t — time series of sampled data

y — sampled values

.returns

f — Frequency of main signal component
A — Amplitude of main signal component
ph — Phase of main signal component

.providesGUF — no

.providesMCM — no

Example

Phase Sensitive Frequency Estimator

Example for algorithm PSFE.

PSFE is an algorithm for estimating the frequency, amplitude, and phase of the fundamental component in harmonically distorted waveforms. The algorithm minimizes the phase difference between the sine model and the sampled waveform by effectively minimizing the influence of the harmonic components. It uses a three-parameter sine-fitting algorithm for all phase calculations. The resulting estimates show up to two orders of magnitude smaller sensitivity to harmonic distortions than the results of the four-parameter sine fitting algorithm.

See also Lapuh, R., "Estimating the Fundamental Component of Harmonically Distorted Signals From Noncoherently Sampled Data," Instrumentation and Measurement, IEEE Transactions on , vol.64, no.6, pp.1419,1424, June 2015, doi: 10.1109/TIM.2015.2401211, URL: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7061456&isnumber=7104190'`

Contents

- Generate sample data
- Call algorithm
- Display results

Generate sample data

Two quantities are prepared: `t` and `y`, representing 1 second of sinus waveform of nominal frequency 100 Hz, nominal amplitude 1 V and nominal phase 1 rad, sampled at sampling frequency 10 kHz.

```
DI = [];
Anom = 1; fnom = 100; phnom = 1;
DI.t.v = [0:1/1e4:1-1/1e4];
DI.y.v = Anom*sin(2*pi*fnom*DI.t.v + phnom);
```

Add noise:

```
DI.y.v = DI.y.v + normrnd(0, 1e-3, size(DI.y.v));
```

Call algorithm

Use QWTB to apply algorithm PSFE to data DI.

```
DO = qwtb('PSFE', DI);
```

```
QWTB: no uncertainty calculation
```

Display results

Results is the amplitude, frequency and phase of sampled waveform.

```
f = DO.f.v
A = DO.A.v
ph = DO.ph.v
```

```
f =

  100.0000


A =

    1.0000
```

```
ph =

     1.0000
```

Errors of estimation in parts per milion:

```
ferrppm = (DO.f.v - fnom)/fnom .* 1e6
Aerrppm = (DO.A.v - Anom)/Anom .* 1e6
pherrppm = (DO.ph.v - phnom)/phnom .* 1e6
```

```
ferrppm =

     0.1322


Aerrppm =

   -10.1106


pherrppm =

   -34.9097
```

4

FPSWF – Four Parameter Sine Wave Fitting

Description

.id — FPSWF

.name — Four Parameter Sine Wave Fit

.desc — Fits a sine wave to the recorded data by means of least squares using 4 parameter model. Different functions are used when run in MATLAB or GNU Octave.

.citation —

.remarks — Algorithm is very sensitive to distortion. Algorithm requires good estimate of frequency.

.license —

.requires

- t — Time series of sampled data
- y — Sampled values

.returns

- f — Frequency of main signal component
- A — Amplitude of main signal component
- ph — Phase of main signal component
- O — Offset of signal

.providesGUF — no

.providesMCM — no

Example

Four parameter sine wave fitting

Example for algorithm FPSWF.

FPSWF is an algorithm for estimating the frequency, amplitude, and phase of the sine waveform. The algorithm use least squares method. Algorithm requires good estimate of frequency.

Contents

- Generate sample data
- Call algorithm
- Display results

Generate sample data

Two quantities are prepared: `t` and `y`, representing 1 second of sinus waveform of nominal frequency 1 kHz, nominal amplitude 1 V, nominal phase 1 rad and offset 1 V sampled at sampling frequency 10 kHz.

```
DI = [];
Anom = 2; fnom = 100; phnom = 1; Onom = 0.2;
DI.t.v = [0:1/1e4:1-1/1e4];
DI.y.v = Anom*sin(2*pi*fnom*DI.t.v + phnom) + Onom;
```

Lets make an estimate of frequency 0.2 percent higher than nominal value:

```
DI.f.v = 100.2;
```

Call algorithm

Use QWTB to apply algorithm FPSWF to data DI.

```
CS.verbose = 1;
DO = qwtb('FPSWF', DI, CS);
```

```
QWTB: no uncertainty calculation
Fitting started

Local minimum found.

Optimization completed because the size of the
   gradient is less than
the default value of the function tolerance.



Fitting finished
```

Display results

Results is the amplitude, frequency and phase of sampled waveform.

```
A = DO.A.v
f = DO.f.v
ph = DO.ph.v
O = DO.O.v
```

```

A =

     2.0000


f =

   100.0000


ph =

     1.0000

```

```
O =

   -0.2000
```

Errors of estimation in parts per milion:

```
Aerrppm = (DO.A.v - Anom)/Anom .* 1e6
ferrppm = (DO.f.v - fnom)/fnom .* 1e6
pherrppm = (DO.ph.v - phnom)/phnom .* 1e6
Oerrppm = (DO.O.v - Onom)/Onom .* 1e6
```

```

Aerrppm =

   4.8894e-07


ferrppm =

  -1.4211e-10


pherrppm =

   1.7542e-08


Oerrppm =

  -2.0000e+06
```

5

SP-FFT – Spectrum by means of Fast Fourier Transform

Description

.id — SP-FFT

.name — Spectrum by means of Fast Fourier Transform

.desc — Calculates frequency and phase spectrum by means of Fast Fourier Transform algorithm. Result is normalized.

.citation —

.remarks —

.license —

.requires

- y — Sampled values
- fs — Sampling frequency

.returns

- f — Frequency series
- A — Amplitude series
- ph — Phase series

.providesGUF — no

.providesMCM — no

Example

Signal Spectrum by means of Fast fourier transform

Example for algorithm SP-FFT.

Calculates frequency and phase spectrum by means of Fast Fourier Transform algorithm. Result is normalized.

Contents

- Generate sample data
- Call algorithm
- Display results

Generate sample data

Two quantities are prepared: y and fs, representing 1 second of signal containing 5 harmonic components and one inter-harmonic component. Main signal component has nominal frequency 1 kHz, nominal amplitude 2 V, nominal phase 1 rad and offset 1 V sampled at sampling frequency 10 kHz.

```
DI = [];
fsnom = 1e4; Anom = 2; fnom = 100; phnom = 1; Onom =
    0.2;
t = [0:1/fsnom:1-1/fsnom];
DI.y.v = Anom*sin(2*pi*fnom*t + phnom);
for i = 2:45
        DI.y.v = DI.y.v + Anom./i*sin(2*pi*fnom*i*t +
    phnom + i - 1);
end
DI.y.v = DI.y.v + 1*sin(2*pi*fnom*1.456*t + phnom);
DI.fs.v = fsnom;
```

Call algorithm

Use QWTB to apply algorithm SP-FFT to data DI.

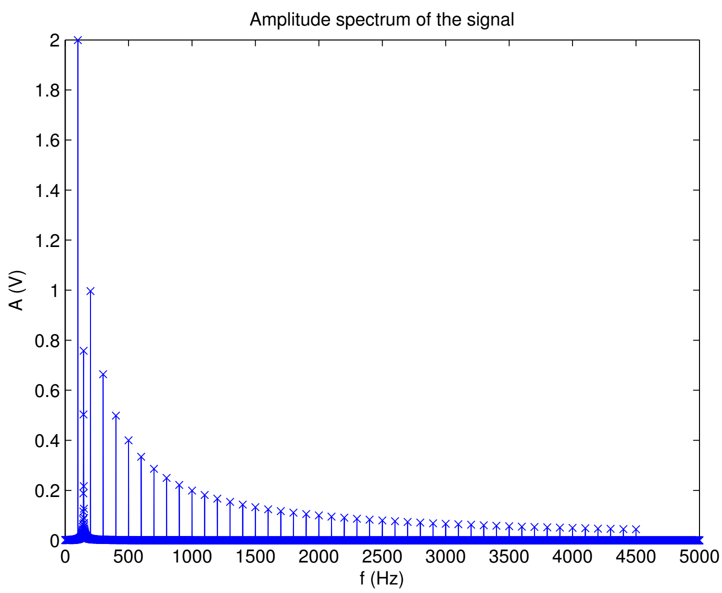
```
DO = qwtb('SP-FFT', DI);
```

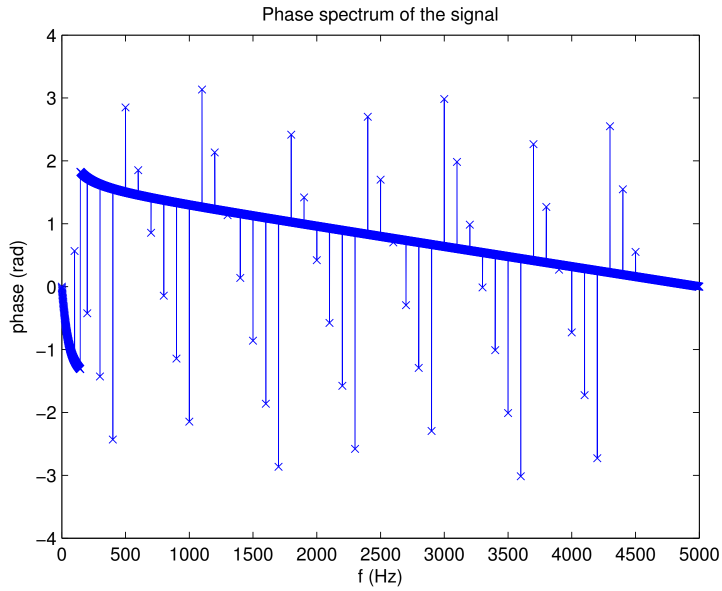
```
QWTB: no uncertainty calculation
```

Display results

Results is the amplitude and phase spectrum.

```
figure
plot(DO.f.v, DO.A.v, '-x')
xlabel('f (Hz)'); ylabel('A (V)'); title('Amplitude
    spectrum of the signal');
figure
plot(DO.f.v, DO.ph.v, '-x')
xlabel('f (Hz)'); ylabel('phase (rad)'); title('Phase
    spectrum of the signal');
```





Phase spectrum of the signal
phase (rad)
f (Hz)
4
3
2
1
0
−1
−2
−3
−4
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000