

# Cyfrowe przetwarzanie sygnałów 2023/2024

## Projekt 2. Próbkowanie i kwantyzacja

Amadeusz Sitnicki, 242524

Adam Rosiak, 242511

Prowadzący: dr hab. inż. Bartłomiej Stasiak

18 kwietnia 2024

### 1 Cel projektu

Celem poprzedniej części zadania było stworzenie programu umożliwiającego generowanie sygnałów zadaną częstotliwością próbkowania oraz implementacja podstawowych operacji pozwalających na generowanie nowego sygnału na podstawie 2 sygnałów bazowych. Dodatkowym celem było stworzenie interfejsu użytkownika oraz wizualizacji w postaci wykresów wraz z podsumowaniem prezentującym podstawowe wartości opisujące sygnał.

Celem bieżącej części zadania jest dodanie do aplikacji funkcjonalności pozwalających na kwantyzację i rekonstrukcję sygnału przy użyciu różnych metod interpolacji. Wpływ parametrów przetwarzania A/C i C/A zostanie zbadany w sekcji analizy wyników (??).

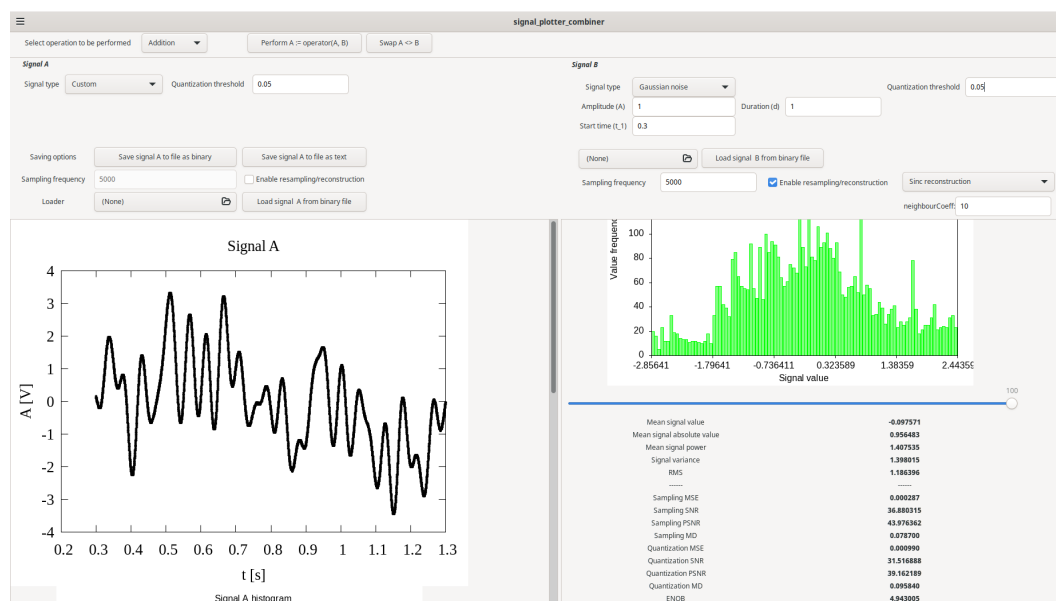
### 2 Instrukcja korzystania z aplikacji

W pierwszej części zadania, użytkownik korzysta z podwójnego menu pozwalającego na modyfikowanie parametrów wybranego typu sygnału - dostępne menu dla sygnału A oraz menu dla sygnału B. Dla obu sygnałów użytkownik korzysta z odpowiedniego dla sygnału okna przewijanego zawierającego w pierwszej kolejności wykres wartości sygnału w zależności od czasu, a następnie histogram prezentujący dla ilu próbek sygnału, jego wartość znajdowała się w określonych przedziałach. Możliwe jest ustawienie z osobna dla obu sygnałów rozdzielczości histogramu przy użyciu suwaka. Poniżej suwaka znajduje się podsumowanie zawierające obliczoną średnią wartość sygnału, średnią wartość bezwzględną sygnału, średnią mocą sygnału, wariancją i wartością skuteczną sygnału.

W bieżącej części zadania, do zawartości okien przewijanych obu sygnałów, dołączyły kontrolki graficznego interfejsu użytkownika, pozwalające na wybór progu kwantyzacji - ustawienie wartości większej od 0, spowoduje skwantyzowanie wybranego sygnału. Obok pola tekstowego przeznaczonego do wpisywania

częstości próbkowania sygnału, dodany został checkbox, pozwalający na przełączanie się między trybem regeneracji sygnału, a trybem rekonstrukcji sygnału z użyciem metod interpolacji. Po aktywacji trybu rekonstrukcji, pojawia się widget rozwijanej listy, pozwalający na wybór dostępnych metod rekonstrukcji spróbkowanego sygnału. W przypadku metody opartej na funkcji *sinc*, konieczne jest ustawienie parametru określającego liczbę sąsiadujących próbek po każdej ze stron punktu na osi czasu, które mają być uwzględniane przy rekonstrukcji sygnału. W trybie rekonstrukcji, każda zmiana częstotliwości próbkowania lub parametru rekonstrukcji, powoduje uruchomienie algorytmu interpolacji, który przekształca sygnał. Dla dużej wyjściowej częstotliwości próbkowania, rekonstruktor/interpolator zachowuje się jak układ DAC przekształcający sygnał dyskretny w analogowy.

Rysunek 1 przedstawia graficzny interfejs użytkownika wraz zawierający wspomniane w tej sekcji elementy interfejsu.



Rysunek 1: Graficzny interfejs użytkownika programu

### 3 Opis implementacji oraz metod generowania danych do wykresów

Aplikacja wykorzystuje otwartoźródłową bibliotekę graficznego interfejsu użytkownika GTK3 [1]. Do implementacji wykorzystano język C wraz z kompilatorem GCC [2] i systemem budowania Meson [3], zapewniającym uproszczoną

integrację z biblioteką GTK3. W celu generowania wykresów, aplikacja posiada interfejs (`_gcall.h`, `gnuplot.h`) dedykowany uruchamianiu procesu `gnuplot` - otwartoźródłowego narzędzia [4] pozwalającego na tworzenie wykresów z wykorzystaniem skryptów w dedykowanym skryptowym języku `gnuplot`. Proces jest uruchamiany z przekazaniem parametrów zależnych od ustawień kontrolowanych przez użytkownika. Wykorzystano programowanie reaktywne - aplikacja reaguje na każdą zmianę parametrów sygnału i aktualizuje wykresy oraz podsumowania. `Gnuplot` pobiera dane generowane poprzez utworzenie przez aplikację pliku danych zawierającego specjalnie sformatowane dane sygnału i przekazanie jego ścieżki jako jeden z argumentów uruchomieniowych procesu `gnuplot`. Przykład zbioru parametrów przekazanych do biblioteki znajduje się poniżej.

```
gnuplot -e "max=1." -e "min=-1." -e "n=10" -e "outfile='foo.png'\"
-e "infile='data.txt'" -e "plottitle='Signal A histogram'" script.plt
```

Plik z danymi (odpowiadający plikowi `data.txt` w powyższym przykładzie) zawiera pary wartości (numer próbki, wartość próbki - oddzielone spacją) po jednej na linię.

Jak wspomniano, aplikacja umożliwia eksport i import sygnału z wykorzystaniem specjalnego formatu zawartości pliku sygnału. Format ten specyfikuje 24 bajty nagłówka, po których następują ośmiobajtowe grupy odpowiadające wartościom sygnału (typ `double`) dla poszczególnych próbek. Poniższe struktury języka C określają wspomniany format zawartości pliku sygnału (por. `signal_fio.h`, `signal.h`)

```
typedef struct {
    double start_time;
    double sampling_frequency;
    uint64_t num_samples;
} signal_info_t;

typedef struct {
    union {
        signal_info_t info;
        uint64_t raw[3];
    } header;
    double* pData;
} real_signal_file_payload_t;
```

Do całokształtu implementacji wykorzystano wzorzec MVC w celu odseparowania warstw modelu, widoku i kontrolera widoku. Warstwa widoku została zaimplementowana z użyciem pliku XML generowanego przez otwartoźródłowe narzędzie `Glade`, dedykowane projektowaniu graficznych interfejsów użytkownika dla biblioteki GTK3. W celu generowania wartości losowej o standardowym rozkładzie normalnym, użyto metody Marsaglia [5], której przykładową, uproszczoną implementację przedstawiono poniżej.

```
/**
```

```

    * Uses Marsaglia polar method and rand() to generate a standard-normally
    distributed pseudo-random floating-point value
    *
    */
double __standard_gaussian_rand() {
    double x; double y;
    double s;
    do {
        x = -1.0 + 2 * ((double)rand()) / (double)RAND_MAX;
        y = -1.0 + 2 * ((double)rand()) / (double)RAND_MAX;
        s = x*x + y*y;
    } while (s >= 1.0);
    return x * sqrt(-2.0 * log(s) / s);
}

```

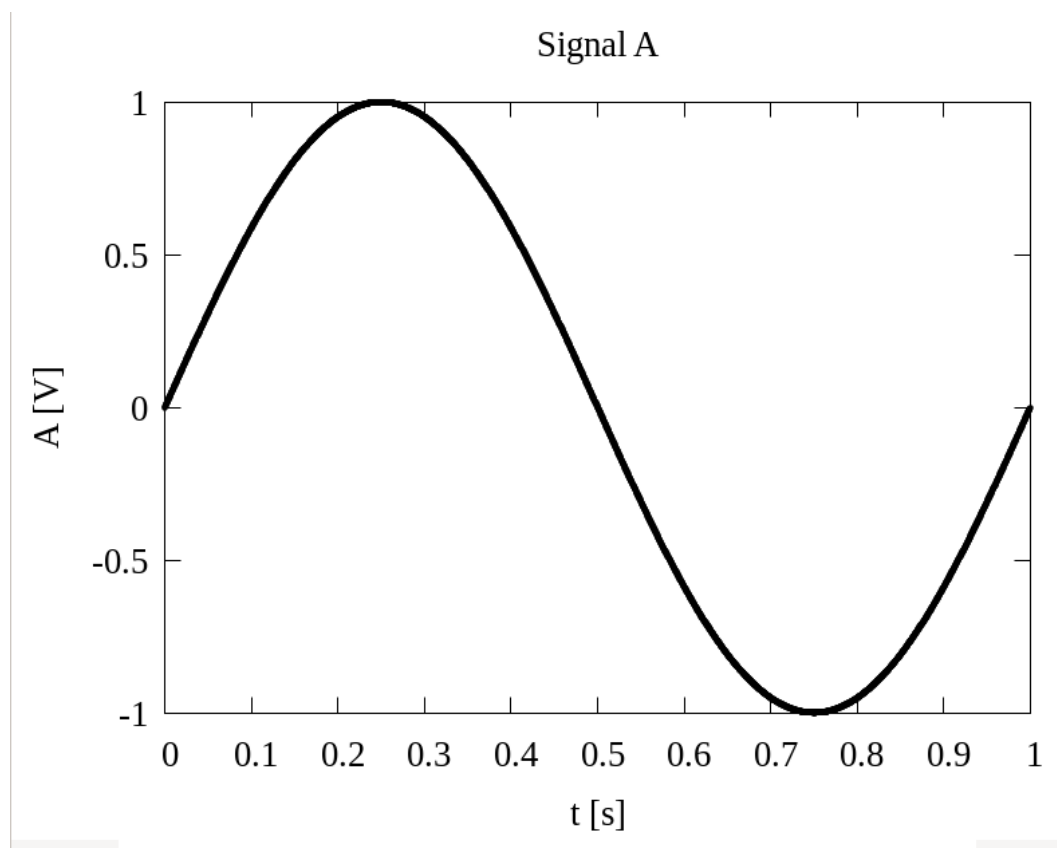
## 4 Przykłady i wnioski z analizy działania programu - część 1

### 4.1 Przykład A - sygnał sinusoidalny

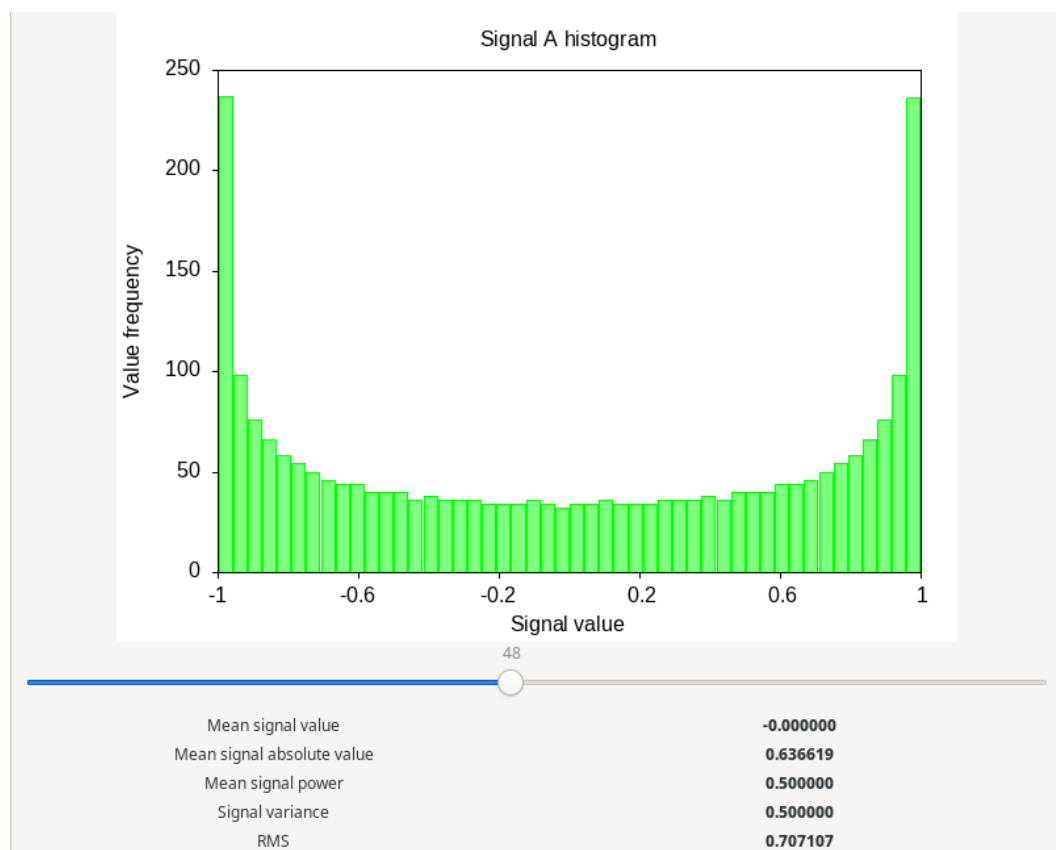
The image shows a software interface for configuring a signal, titled "Signal A". The interface includes several input fields and buttons:

- Signal type:** A dropdown menu set to "Sine".
- Amplitude (A):** A text input field containing the value "1".
- Start time (t<sub>1</sub>):** A text input field containing the value "0".
- Period (T):** A text input field containing the value "1".
- Duration (d):** A text input field containing the value "1".
- Saving options:** Two buttons: "Save signal A to file as binary" and "Save signal A to file as text".
- Sampling frequency:** A text input field containing the value "2560".
- Loader:** A dropdown menu set to "(None)" with a folder icon, and a button "Load signal A from binary file".

Rysunek 2: Konfiguracja dla przykładowego sygnału sinusoidalnego



Rysunek 3: Wykres wartości sygnału sinusoidalnego



Rysunek 4: Histogram i wartości agregatów dla sygnału sinusoidalnego

## 4.2 Wniosek do przykładu A

Skrajne wartości sygnału występują wśród próbek sygnału częściej, jest to spowodowane faktem, że w pobliżu ekstremów, funkcja sinus zmienia się wolniej, zatem jest więcej próbek przypadających na zadany przedział wartości. Dodatkowo warto zwrócić uwagę, że średnia wartość sygnału sinusoidalnego na przedziale o długości będącej wielokrotnością długości okresu, jest równa 0. Wariancja wartości sygnału oraz średnia moc sygnału mają wartość 0,50. Wartość skuteczna sygnału jest odwrotnością pierwiastka kwadratowego liczby 2.

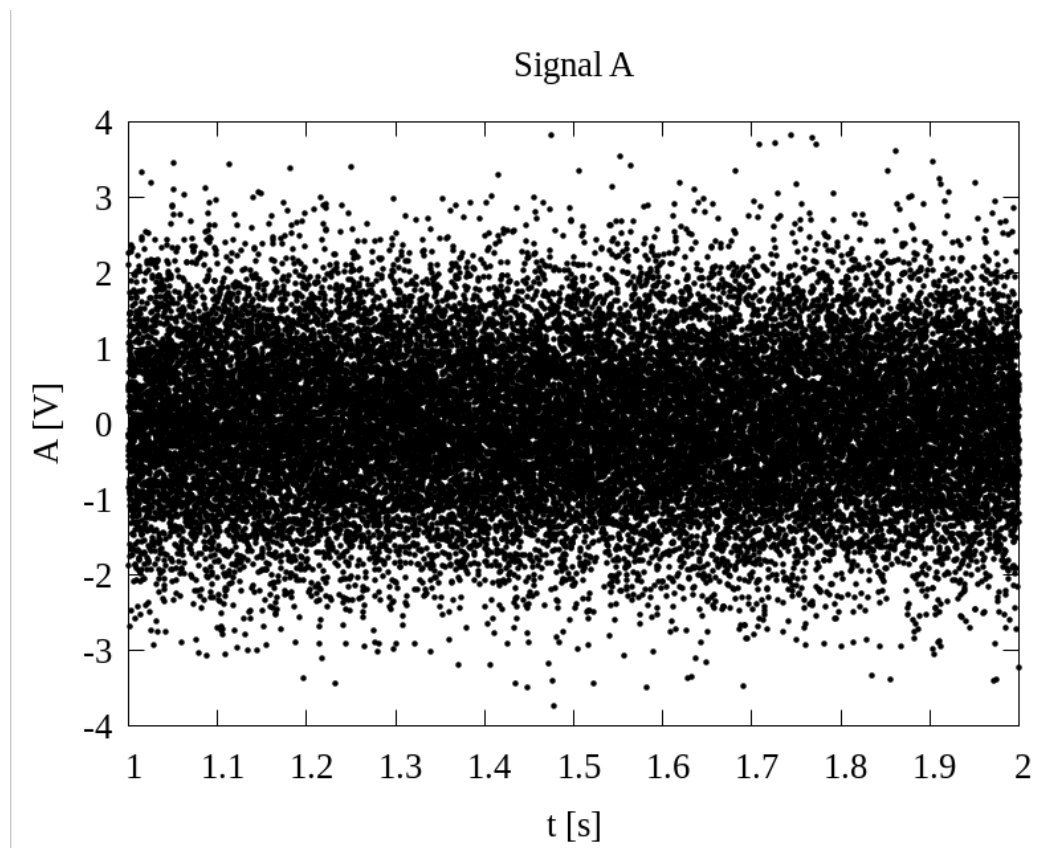
## 4.3 Przykład B - szum gaussowski

The image shows a web-based configuration interface for a signal named "Signal A". The interface includes several input fields and buttons. The "Signal type" is set to "Gaussian noise". The "Amplitude (A)" is set to 1, and the "Duration (d)" is set to 1. The "Start time (t\_1)" is set to 1. Under "Saving options", there are two buttons: "Save signal A to file as binary" and "Save signal A to file as text". The "Sampling frequency" is set to 25600. Under "Loader", there is a button labeled "(None)" with a folder icon, and a button labeled "Load signal A from binary file".

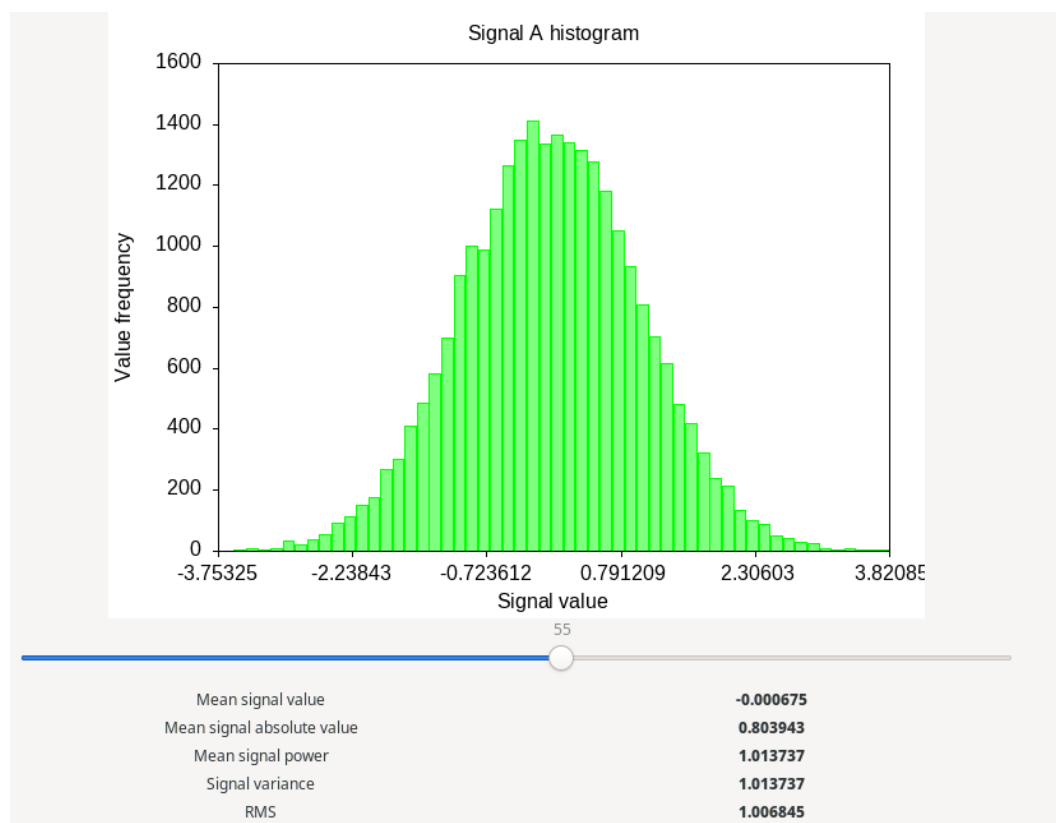
Signal A	
Signal type	Gaussian noise
Amplitude (A)	1
Duration (d)	1
Start time (t_1)	1
Saving options	
<input type="button" value="Save signal A to file as binary"/> <input type="button" value="Save signal A to file as text"/>	
Sampling frequency	25600
Loader	
<input type="button" value="(None)"/> <input type="button" value="Load signal A from binary file"/>	

Rysunek 5: Konfiguracja dla standardowego szumu gaussowskiego





Rysunek 6: Wykres wartości standardowego szumu gaussowskiego

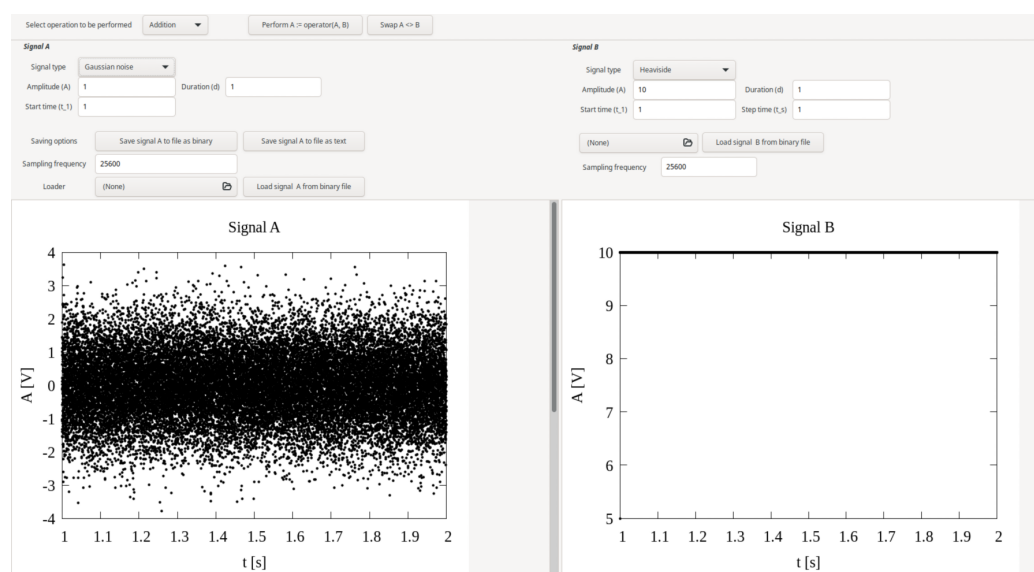


Rysunek 7: Histogram i wartości agregatów dla standardowego szumu gaussowskiego

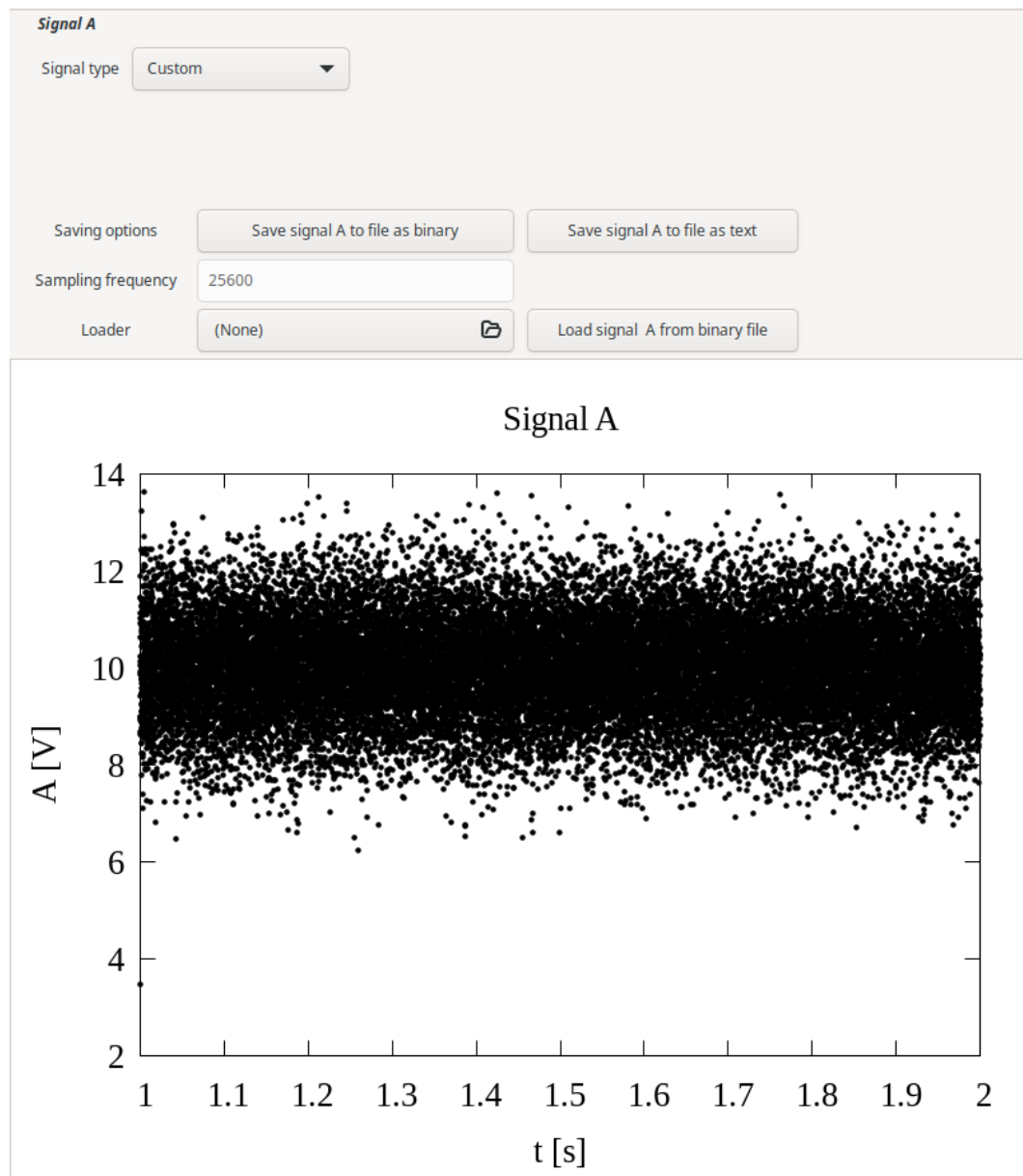
#### 4.4 Wniosek do przykładu B

Metoda polarna Marsaglia jest skutecznym sposobem na otrzymanie przybliżonego standardowego rozkładu normalnego pseudo-losowej wartości zmienno-przecinkowych. Wartość skuteczna szumu jest bliska 1. Zgodnie z definicją użytego rozkładu, wartość oczekiwana dla sygnału jest bliska 0, a wariancja wynosi w przybliżeniu 1.

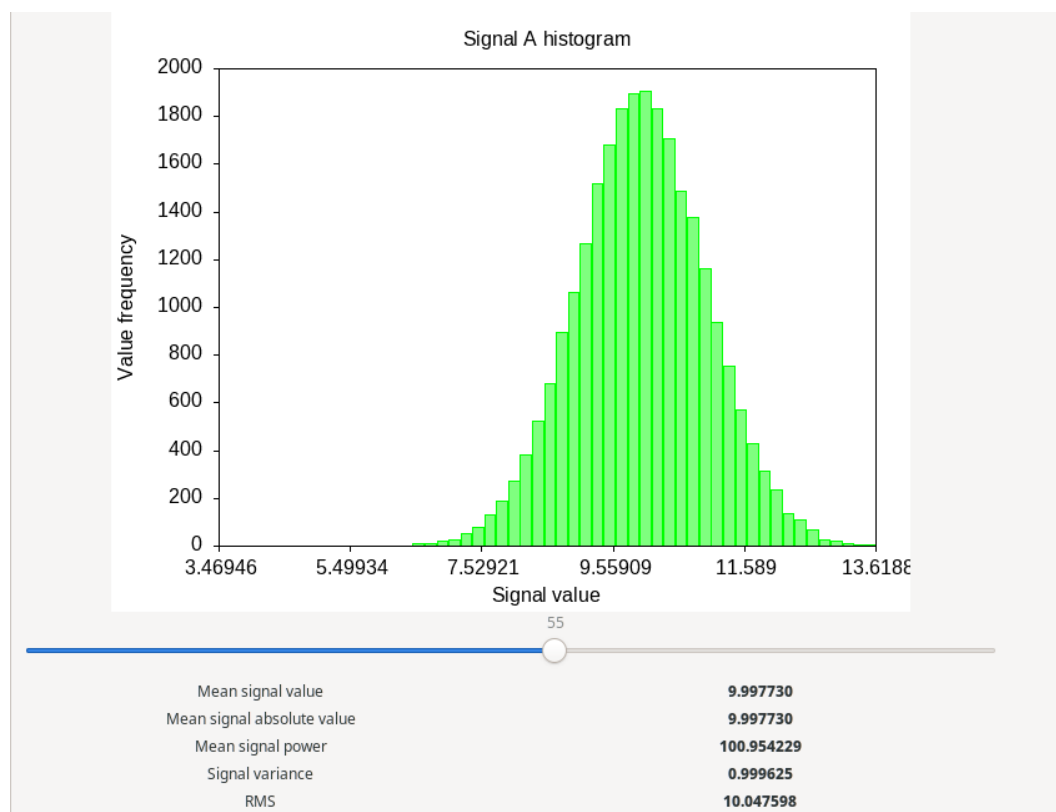
#### 4.5 Przykład C - kombinacja sygnałów z użyciem prostej operacji dodawania



Rysunek 8: Konfiguracja dla dodawania sygnału o stałej wartości 10 do szumu gaussowskiego



Rysunek 9: Wykres wartości sygnału wynikowego operacji dodawania - przesunięty standardowy szum gaussowski



Rysunek 10: Histogram dla sygnału wynikowego operacji dodawania - przesunięty standardowy szum gaussowski

## 4.6 Wniosek do przykładu C

Dodanie dodatniego sygnału stałego do szumu gaussowskiego powoduje przesunięcie wykresu na histogramie w prawo - operacja dodawania zachodzi pomyślnie. Zaobserwować można ustalenie się wartości skutecznej oraz średniej wartości sygnału i średniej absolutnej wartości sygnału na poziomie o 10 wyższym niż w przypadku standardowego rozkładu gaussowskiego bez przesunięcia - liniowa zmiana względem przesunięcia wartości sygnału. Wariancja sygnału nie uległa zmianie względem szumu bez przesunięcia, zaś średnia moc wzrosła o kwadrat przesunięcia wartości szumu.

## 5 Przykłady i wnioski z analizy działania programu - część 2

W ramach przykładu, analizy dokonano dla funkcji będącej sumą 2 funkcji sinusoidalnych o różnych częstotliwościach:  $f(x) = \sin(2\pi t) + \sin(8\pi t)$ . Przyjęto przybliżenie sygnału analogowego z wykorzystaniem częstotliwości próbkowania 250000.

### 5.1 Analiza - próbkowanie

Wykorzystano interpolację zero-hold, a także rekonstrukcję z wykorzystaniem funkcji *sinus cardinalis*.

#### 5.1.1 Interpolacja zero-hold

$f_s$	$MSE$	$SNR$	$PSNR$	$MD$
25	0.192	7.165	12.681	1.139
50	0.053	12.775	18.360	0.579
75	0.048	13.151	18.847	0.413
100	0.013	18.974	24.675	0.295
200	0.003	24.905	30.606	0.157

Tabela 1: Miary podobieństwa sygnału dyskretnego i sygnału zrekonstruowanego dla różnych wartości częstotliwości próbkowania  $f_s$ .

### 5.1.2 Rekonstrukcja z użyciem funkcji *sinc*

$f_s$	<i>sincCountCoeff</i>	<i>MSE</i>	<i>SNR</i>	<i>PSNR</i>	<i>MD</i>
25	5	0.000	88.107	93.623	0.000
25	10	0.000	88.202	93.718	0.000
50	5	0.000	90.493	96.078	0.000
50	10	0.000	89.120	94.705	0.000

Tabela 2: Miary podobieństwa sygnału dyskretnego i sygnału zrekonstruowanego dla różnych wartości częstotliwości próbkowania  $f_s$ .

## 5.2 Analiza - kwantyzacja

Wykorzystano kwantyzację równomierną z obcięciem.

## Literatura

- [1] GTK3, strona projektu, <https://docs.gtk.org/gtk3/>
- [2] GCC, strona projektu, <https://gcc.gnu.org/>
- [3] Meson Build, strona projektu, <https://mesonbuild.com/>
- [4] Gnuplot, strona główna projektu, <http://www.gnuplot.info/>
- [5] Marsaglia G., Bray T.A. (1964), A Convenient Method for Generating Normal Variables

Literatura zawiera wyłącznie źródła recenzowane i/lub o potwierdzonej wiarygodności, możliwe do weryfikacji i cytowane w sprawozdaniu.