リスト1　GNU Radio本体やRTL-SDR関連ソフトウェア，開発環境のインストール

```
$ sudo apt update; sudo apt upgrade; sudo apt autoremove
$ sudo apt install gnuradio gr-osmosdr rtl-sdr gnuradio-dev libspdlog-dev
clang-format xterm git build-essential gdb cmake
```

リスト2　lib/double_complex_impl.hの修正内容

```
省略
namespace gr {
  namespace mymodule {
    class double_complex_impl : public double_complex
    {
     private:
      float _factor;
     public:
      double_complex_impl(float factor);
      ~double_complex_impl();
      // Where all the action really happens
      int work(
              int noutput_items,
              gr_vector_const_void_star &input_items,
              gr_vector_void_star &output_items
      );
    };
  } // namespace mymodule
} // namespace gr
```

リスト3　lib/double_complex_impl.ccの修正内容

```
省略
namespace gr {
  namespace mymodule {
    using input_type = gr_complex;
    using output_type = gr_complex;

    double_complex::sptr
    double_complex::make(float factor)
    {
      return gnuradio::make_block_sptr<
                                double_complex_impl>(
        factor);
```

```
    }
    /*
     * The private constructor
     */
    double_complex_impl::double_complex_impl(float
                                             factor)
      : gr::sync_block("double_complex",
          gr::io_signature::make(1 /* min inputs */,
          1 /* max inputs */, sizeof(input_type)),
          gr::io_signature::make(1 /* min outputs */,
          1 /*max outputs */, sizeof(output_type)))
    {
      _factor = factor;
    }
省略
    int
    double_complex_impl::work(int noutput_items,
        gr_vector_const_void_star &input_items,
        gr_vector_void_star &output_items)
    {
      auto in = static_cast<const input_type*>(
                                      input_items[0]);
      auto out = static_cast<output_type*>(
                                     output_items[0]);
      // Do <+signal processing+>
      for(int index=0; index < noutput_items; index++)
      {
        out[index] = in[index] * _factor;
      }
      return noutput_items;
    }
  } /* namespace mymodule */
} /* namespace gr */
```

リスト4　grc/mymodule_double_complex.block.ymlの修正内容

```
id: mymodule_double_complex
label: double_complex
category: '[mymodule]'

templates:
  imports: from gnuradio import mymodule
  make: mymodule.double_complex(${factor})
```

```
parameters:
- id: factor
  label: Multiply factor
  dtype: float
  default: 1.0

inputs:
- label: in
  domain: stream
  dtype: complex

outputs:
- label: out
  domain: stream
  dtype: complex

file_format: 1
```

リスト5　独自ブロックをインストール

```
$ rm -fr build
$ mkdir build
$ cd build
$ cmake ../
$ make
$ sudo make install
$ sudo ldconfig
```

リスト6　モジュールのアンインストール

```
$ sudo make uninstall
```

```
Creating Python OOT with gr-modtool
https://wiki.gnuradio.org/index.php?title=Creating_Python_OOT_with_gr-
modtool

Creating c++ OOT with gr-modtool
https://wiki.gnuradio.org/index.php?title=Creating_c%2B%2B_OOT_with_gr-
modtool
```

```
% Step 0 元の m スクリプトソース
% see also https://jp.mathworks.com/help/coder/gs/
%                                  averaging-filter.html
close all;
f = 10;
v = 0:0.00614*f:2*pi*f;
x_i = cos(v) + 0.3*rand(1,numel(v));
x_q = sin(v) + 0.3*rand(1,numel(v));
x = x_i + 1i .* x_q;
figure(1);
hold off;
plot(real(x), 'red');

% Use a persistent variable 'buffer' that represents
% a sliding window of s samples at a time.
s = 16; % num of average
buffer = zeros(s,1);

y = zeros(size(x), class(x));
for i = 1:numel(x)
    % Scroll the buffer
    buffer(2:end) = buffer(1:end-1);
    % Add a new sample value to the buffer
    buffer(1) = x(i);
    % Compute the current average value of the window
    % and write result
    y(i) = sum(buffer)/numel(buffer);
end

% Plot the result when the MEX function is applied to
% the noisy sine wave.
% The 'hold on' command ensures that the plot uses the
% same figure window as the previous plot command.
hold on;
plot(real(y), 'blue');

% FFT (dB)
hx = 20*log10(abs(fft(x)));
hy = 20*log10(abs(fft(y)));

figure(2);
hold off;
```

```
plot(hx);
hold on;
plot(hy);
```

リスト8　作成したaveraging_fft.m

```
% Step 1 更新した m スクリプト・ソース
% see also https://jp.mathworks.com/help/coder/gs/averaging-filter.html
close all;
f = 10;
v = 0:0.00614*f:2*pi*f;
x_i = cos(v) + 0.3*rand(1,numel(v));
x_q = sin(v) + 0.3*rand(1,numel(v));
x = x_i + 1i .* x_q;
figure(1);
hold off;
plot(real(x), 'red');
s = 16;

hy = averaging_fft(x,s);

% FFT (only x)
hx = 20*log10(abs(fft(x)));

figure(2);
hold off;
plot(hx);
hold on;
plot(hy);
```

リスト9　averaging_fft関数を使うように改造したスクリプト・ソース

```
% hy = averaging_fft(x,s)
function hy = averaging_fft(x,s) %#codegen
% Use a persistent variable 'buffer' that represents a
% sliding window of s samples at a time.
buffer = zeros(s,1);

y = zeros(size(x), class(x));
for i = 1:numel(x)
    % Scroll the buffer
    buffer(2:end) = buffer(1:end-1);
```

```
    % Add a new sample value to the buffer
    buffer(1) = x(i);
    % Compute the current average value of the window
    % and write result
    y(i) = sum(buffer)/numel(buffer);
end

hy = 20*log10(abs(fft(y)));
```

リスト10　mymatlabモジュールの作成とブロック名avefftを追加

```
$ cd ~
$ gr_modtool newmod mymatlab
$ cd gr-mymatlab
$ gr_modtool add avefft -t sync -l cpp

Please specify the copyright holder:
Enter valid argument list, including default arguments:
 int s = 16
Add Python QA code? [Y/n] Y
Add C++ QA code? [y/N] y
```

リスト11　平均化回数のパラメータを保持する_sをprivateに定義

```
namespace gr {
  namespace mymatlab {
    class avefft_impl : public avefft
    {
      private:
        int _s;
```

リスト12　MATLAB Coderのヘッダ・ファイルをインクルードする

```
#include <gnuradio/io_signature.h>
#include "avefft_impl.h"
#include <iostream>
#include "averaging_fft.h"
#include "averaging_fft_emxAPI.h"
#include "averaging_fft_emxutil.h"
#include "averaging_fft_initialize.h"
#include "averaging_fft_terminate.h"
```

リスト13　namespace以下を編集

```
namespace gr {
  namespace mymatlab {
    using input_type = gr_complex;
    using output_type = float;

    emxArray_creal_T *x;
    emxArray_real_T *hy;
```

リスト14　引数sを_sに保持させる

```
    avefft_impl::avefft_impl(int s)
      : gr::sync_block("avefft",
          gr::io_signature::make(1 /* min inputs */,
          1 /* max inputs */, sizeof(input_type)),
          gr::io_signature::make(1 /* min outputs */,
          1 /*max outputs */, sizeof(output_type)))
    {
      _s = s;
    }
```

リスト15　GNU Radio の実行時に呼ばれるworkメソッド

```
int avefft_impl::work(int noutput_items,
    gr_vector_const_void_star &input_items,
    gr_vector_void_star &output_items)
  {
    auto in = static_cast<const input_type*>(
                                      input_items[0]);
    auto out = static_cast<output_type*>(
                                     output_items[0]);
   if(noutput_items > 1024)
    {
      noutput_items = 1024;
    }else{
      if(noutput_items < 1024)
      {
        noutput_items = 0;
        return 0;
```

```
    }
  }
  x = emxCreate_creal_T(1,noutput_items);
  hy = emxCreate_real_T(1,noutput_items);
  // copy from input
  for(int index = 0; index < noutput_items;index++)
  {
    x->data[index].re = (double)(in[index].real());
    x->data[index].im = (double)(in[index].imag());
  }
  // execute algorithm
  averaging_fft(x, _s, hy);
  // copy to output
  for(int index = 0; index < noutput_items;index++)
  {
    out[index]=(float)(hy->data[index]);
  }
  emxDestroyArray_creal_T(x);
  emxDestroyArray_real_T(hy);
  // Tell runtime system how many output items
  // we produced.
  return noutput_items;
}
```

リスト16　CMakeLists.txtの追加部分

```
########################################
# Setup library
########################################
include(GrPlatform) #define LIB_SUFFIX
list(APPEND mycoder_sources
    mcoder_avefft_impl.cc
    averaging_fft.c
    averaging_fft_emxAPI.c
    averaging_fft_emxutil.c
    averaging_fft_data.c
    averaging_fft_initialize.c
    averaging_fft_terminate.c
    FFTImplementationCallback.c
    rt_nonfinite.c
    rtGetInf.c
    rtGetNaN.c
)
```

リスト17　.yml の編集

```
id: mycoder_mcoder_avefft
label: mcoder_avefft
category: '[mycoder]'
templates:
  imports: from gnuradio import mycoder
  make: mycoder.mcoder_avefft(${s})

parameters:
- id: s
  label: Num of average
  dtype: int
  default: 16

inputs:
- label: in
  domain: stream
  dtype: complex

outputs:
- label: out
  domain: stream
  dtype: float

file_format: 1
```

リスト18　ビルドとインストール

```
$ rm -fr build
$ mkdir build
$ cd build
$ cmake ../
$ make
$ sudo make install
$ sudo ldconfig
```