VECTOR

FFT-Analysis with CANape

2022-09-19
Support Note SN-IMC-1-004

Author(s) Sundt, Alexander; Sokratov, Stanislav

Table of Contents

1 Overview 2
1.1 Advantages of using a DLL 2
1.2 Where to find the DLL file with the FFT functionality? 2
1.3 How to link the DLL LibraryDLL.dll in CANape? 2
1.4 How to use LibraryDll.dll to compute an FFT analysis of a measured signal? 3
1.5 Function code for offline evaluation of a FFT analysis 12
1.6 How to use the CANape demo project 14
1.7 Associated Files 14
2 Contacts 14

1 Overview

There are many ways to compute a FFT (Fast Fourier Transform) analysis with CANape. This Support Note describes a method using a function DLL.

1.1 Advantages of using a DLL

There are two main advantages to this method compared to others:

- this method can be applied to offline and online analysis of data,
- the method is down-gradable to previous CANape versions.

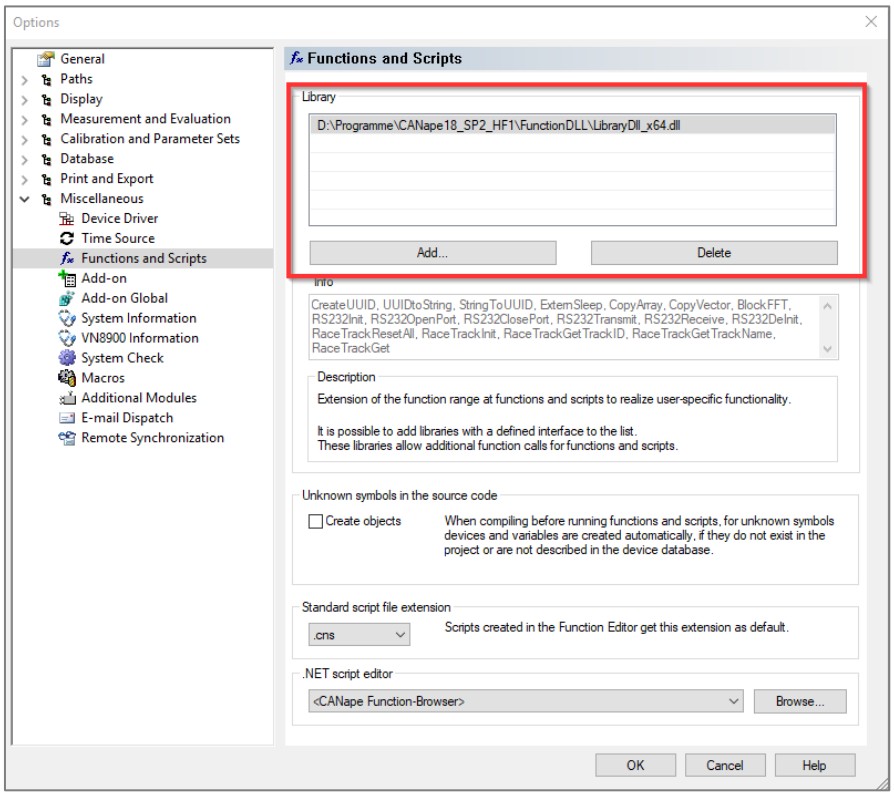
You just need the function DLL, which is already part of CANape since CANape version 12. If you are using an older version of CANape please contact the Vector Support Team. They will be pleased to send you the DLL file.

1.2 Where to find the DLL file with the FFT functionality?

In the directory of your CANape installation you find a folder named `FunctionDLL`. Within this folder you find a DLL file called `LibraryDll.dll`.

1.3 How to link the DLL LibraryDLL.dll in CANape?

Start CANape and open the CANape options (**Tools | Options**). Open the tab **Miscellaneous** and select **Functions and Scripts**. On the right-hand side of the dialogue **Options** you find a column listing all DLL files that are already included to your CANape project. By default this column is empty. Add the DLL `LibraryDll.dll` to your project by clicking the **Add** button and selecting the DLL file `LibraryDLL.dll` in your CANape installation directory. Confirm and close the dialogue **Options.**



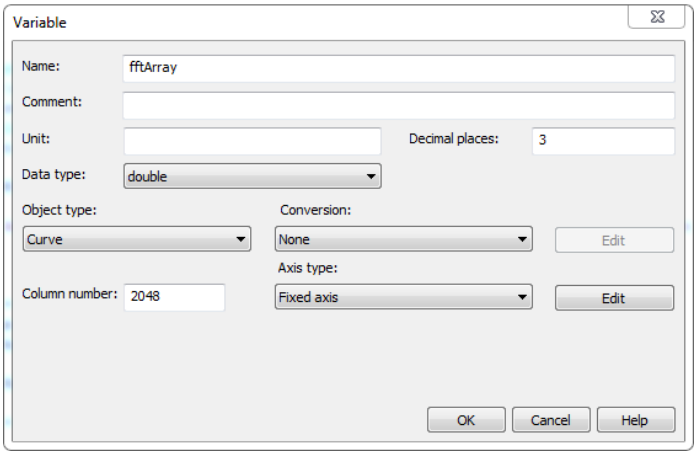
1.4 How to use LibraryDll.dll to compute an FFT analysis of a measured signal?

The `LibraryDLL.dll` includes a scripting function to calculate an FFT called 'BlockFFT'. The following example describes the use of this function for channel 1 of 'XCPsim' simulator:

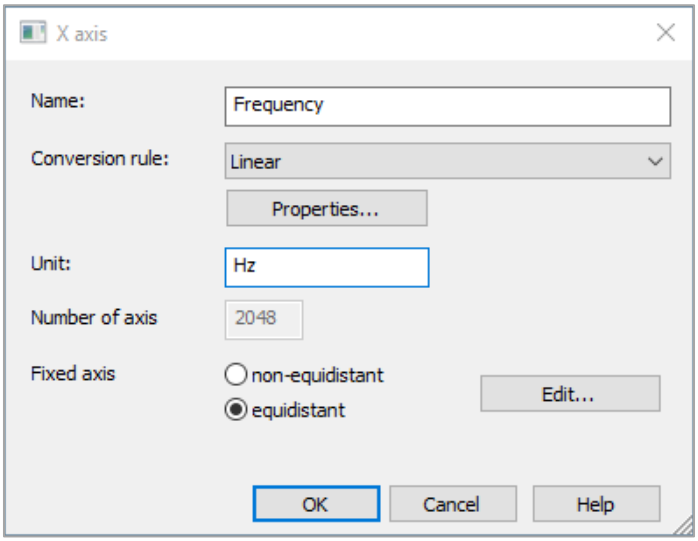
1. The results of the FFT calculation will be saved to a global variable. Hence, create a global variable of data type **double** and object type **curve** and name it e.g. 'fftArray'. By setting the number of columns you define the linewidth of your FFT-spectrum.

> **Note**
> The column number has to be a power of two and less than 4096!

Set the axis type to **fixed axis**.



Later on you will want to visualize the calculated FFT. Typically the x-axis will be labeled **Frequency** in units of **Hz** or **kHz**. Click on the **Edit** button next to the axis type settings. The new dialog allows you to set the x-axis label (here **Frequency**) and the unit (here **Hz**). To translate the scaling of the x-axis into frequency set the conversion rules to 'linear' and click on the **Properties** button.

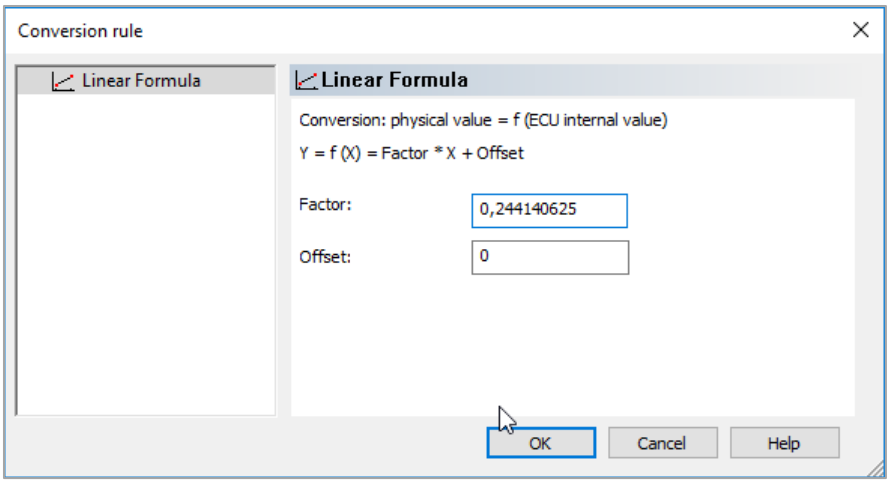


Set the conversion factor to match your measurement: Sampling rate of the signal divided by two times the number of columns of this array.

Example

Channel 1 is sampled with a sampling rate of 1000 Hz (measurement mode = 1ms) and the array 'fftArray' has 2048 columns.

The factor calculates to 1000 Hz / (2 * 2048) = 0,244140625



2. Write a project function. This article provides a function code to online computing a normalized power spectrum over the last 4000 measured data points (BLOCK_SIZE) and a function code for offline computation of the FFT at the very end of this article:

Note

The function code *Funktion_DoTheFFT_online* is available in FFT_Analysis.zip.

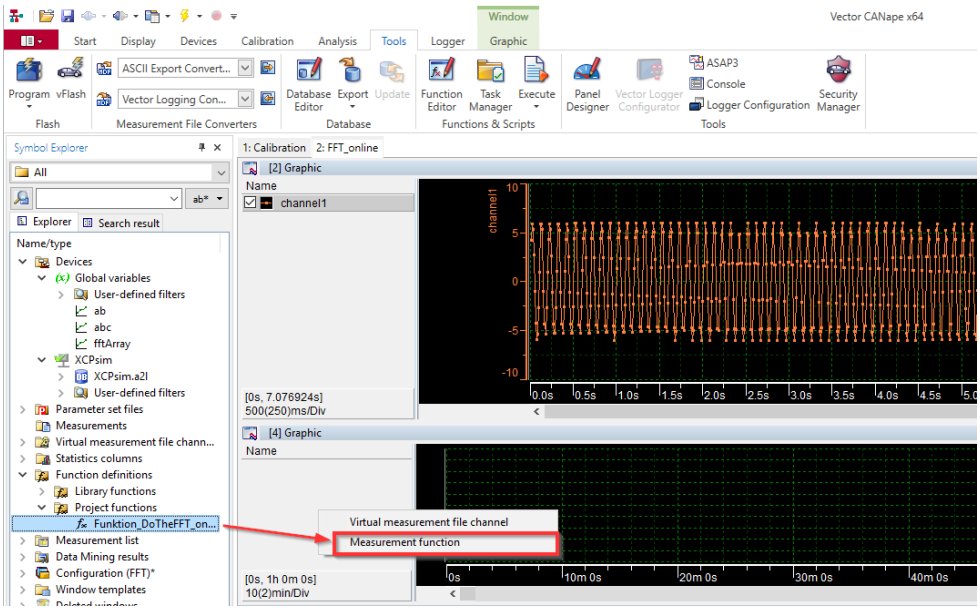
```
/*****************************************************************************
this function calculates the FFT over the previous BUFFER_SIZE number of
data points.

Within this function six parameters are defined (with R and I denote the
real and imaginary part of the FFT):

BUFFER_SIZE = defines the number of data points that are taken into account
by the function BlockFFT. If the number of datapoints previous to the
current analyzed data point is less than BUFFER_SIZE then the missing
parameter values for BlockFFT are omitted.

    BLOCK_FFT_FLAG_NORMALIZE = flag to normalize the result of BLOCK_FFT
    BLOCK_FFT_MODE_PEAK_AMPLITUDE = ( R^2 + I^2 )^0.5
    BLOCK_FFT_MODE_RMS_AMPLITUDE = ( ( R^2 +I^2 )^0.5 ) / ( 2 )^0.5
    BLOCK_FFT_MODE_AUTO_SPECTRUM = R^2 + I^2
    BLOCK_FFT_MODE_POWER_SPECTRUM = ( R^2 + I^2 ) / 2
*****************************************************************************
********/

function Funktion_DoTheFFT_online (var signal, var fftArray[])
{
/********************** DEFINE PARAMETERS *********************************/
  #define BUFFER_SIZE   4000
  #define BLOCK_FFT_FLAG_NORMALIZE               0x0001
// Flag, can/must be combined with a mode
  #define BLOCK_FFT_MODE_PEAK_AMPLITUDE          0x0002
  #define BLOCK_FFT_MODE_RMS_AMPLITUDE           0x0004
  #define BLOCK_FFT_MODE_AUTO_SPECTRUM           0x0008
```

```
  #define BLOCK_FFT_MODE_POWER_SPECTRUM           0x0010


/************************ LOCAL VARIABLES ******************************/
  double buffer[BUFFER_SIZE];
  double n;
  double dim;
  double checkVal;
  double dt;
  double t_start;
  double t_end;
  long initial = 1;
  long idx;
  long raster;
  long i = -1;
  long isPowerOfTwo;


/************************ MAIN CODE ************************************/

  // ----------initialize variables----------------
  switch (initial)
  {
    case 1:  // validate fftArray settings
      initial = 2;

      // check if dimension of fftArray is less than 4096
      dim = xdimension(fftArray);
      if(dim >= 4096)
      {
        Write("The number of columns of fftArray is too large (currently
%d). The calculation cannot be computed and is stopped. Reduce the number
of columns to a value less than 4096. NOTE: The number of columns has to be
a power of two. ", dim);
        cancel;
      }

      // check if dimension of fftArray is a power of two
      isPowerOfTwo = 1;
      checkVal = dim;
      while((checkVal != 1) && (isPowerOfTwo == 1))
      {
        if(checkVal%2 == 0)
        {
          checkVal = checkVal / 2;
        }else{
          isPowerOfTwo = 0;
        }
      }
      if(isPowerOfTwo == 0)
      {
        Write("The number of columns of fftArray is not a power of two
(currently %d). The calculation cannot be computed and is stopped. Adjust
the number of columns.", dim);
        cancel;
      }

      // initialize global array fftArray, all elements equal to zero
      for(idx = 0; idx < dim; idx++)
      {
        fftArray[idx] = -1;
```

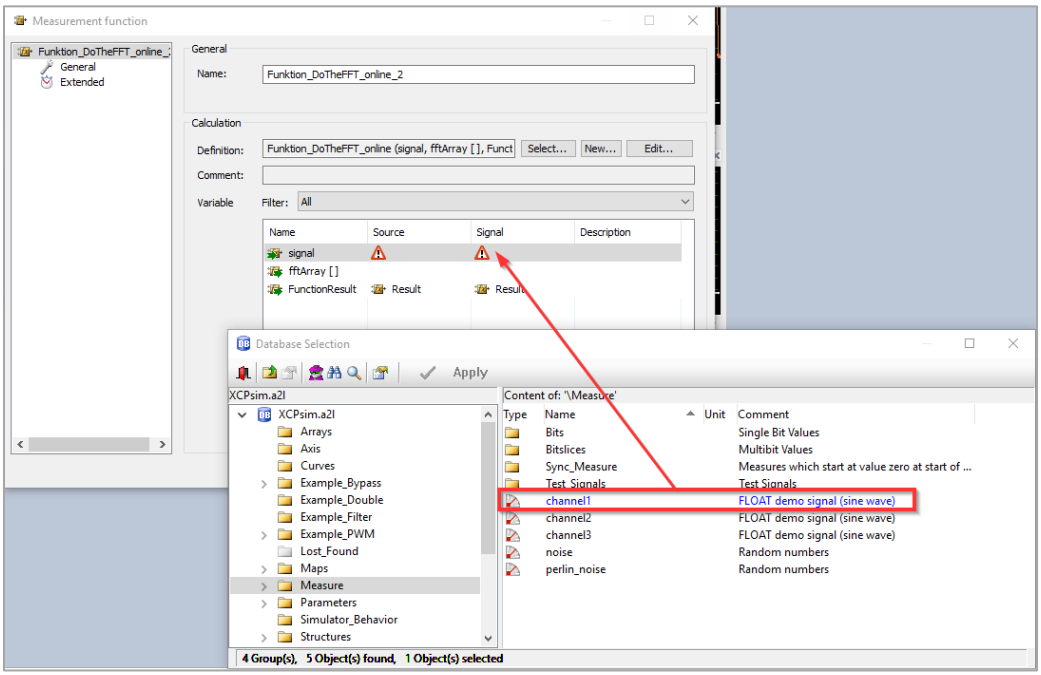
```
      }
      break;

    case 2:   // initialize time raster to calculate the FFT. NOTE:
increasing BUFFER_SIZE automatically increases the calculation time for the
FFT. Hence, the time raster to repeat the FFT calculation is increased.
Otherwise data loss may occur.
      initial = 0;
      dt = diffTime(signal) * 1000;
      raster = ceil(1 / dt);
      i = 1;
      break;
  }

  if (i == raster)
  {
    i = 1;
    t_start = GetClockHR();

    // copy the last BUFFER_SIZE number of data points to the array buffer
to be FFT analyzed.
    for (n = 0; n < BUFFER_SIZE; n++)
    {
      buffer[n] = signal[-n];
    }
    //calculate FFT
    BlockFFT(buffer, BUFFER_SIZE, BLOCK_FFT_FLAG_NORMALIZE |
BLOCK_FFT_MODE_POWER_SPECTRUM, fftArray);

    t_end = GetClockHR();


    // check if raster is sufficient to calculate the FFT. If not adjust
raster setting.
    if ( ((t_end - t_start) / dt) > raster )
    {
      raster = ceil( (t_end - t_start + 1) / dt);
    }

    return t_end - t_start;

  }

  i++;

  return;
}
/********************************** END **********************************
```

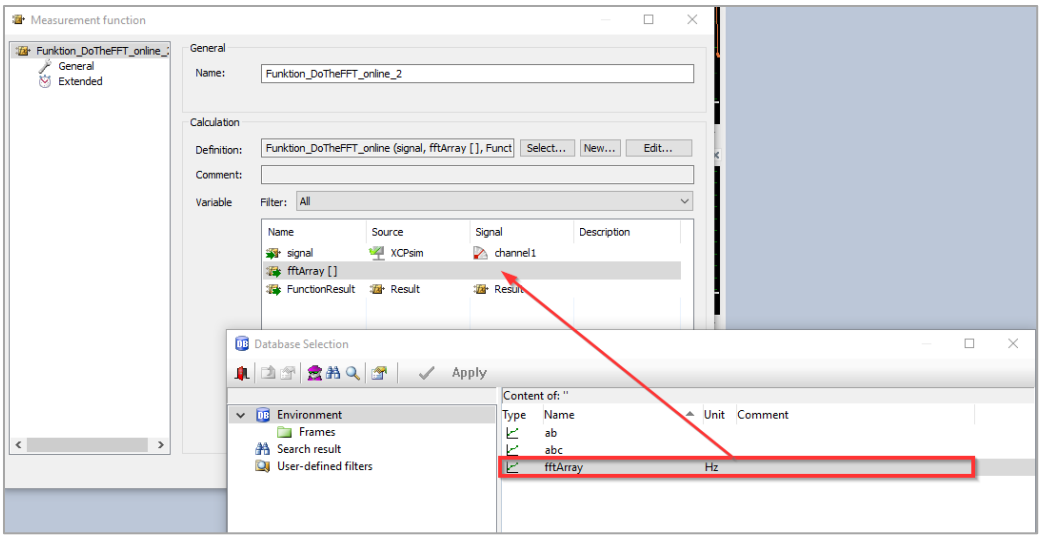
3. Save and close the function generator.

4. Create a measurement function using the project function ‘DoTheFFT_online’, e.g. by drag and dropping the project function from the symbol explorer into a graphic window. Include the project function as a measurement function.



5. Link the signal and the global array to the measurement function, e.g. by using drag and drop functionality.

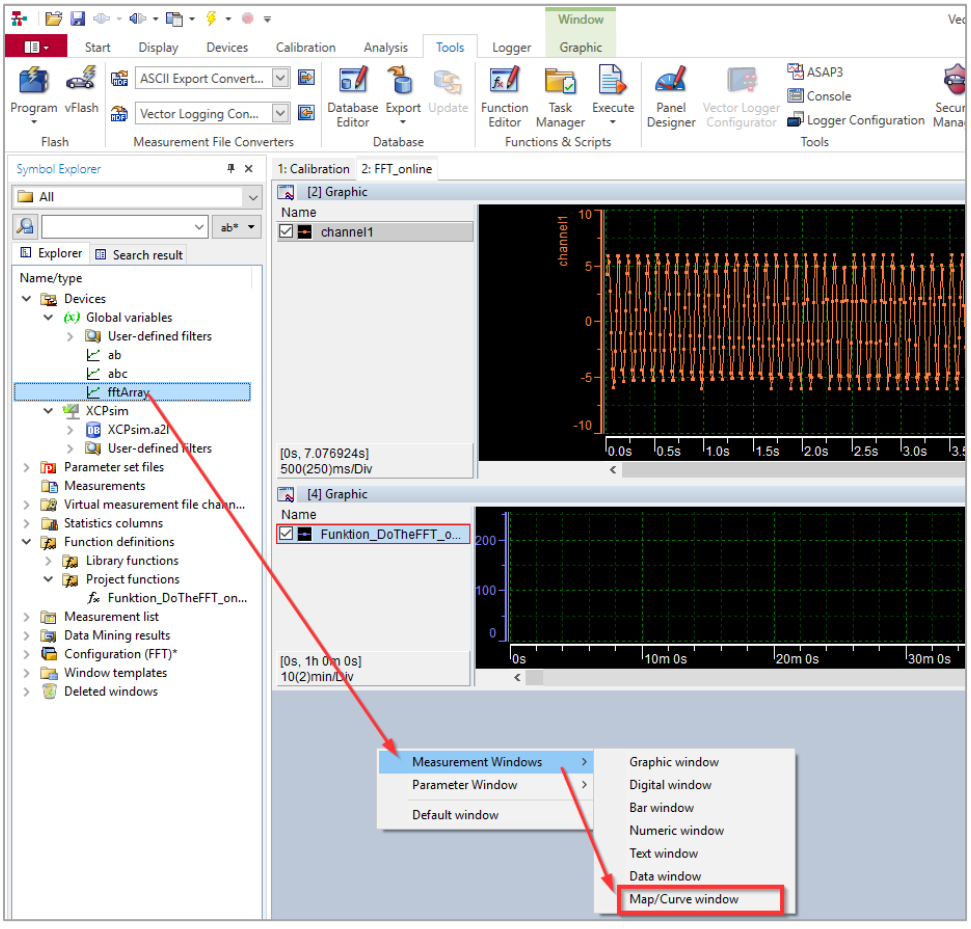




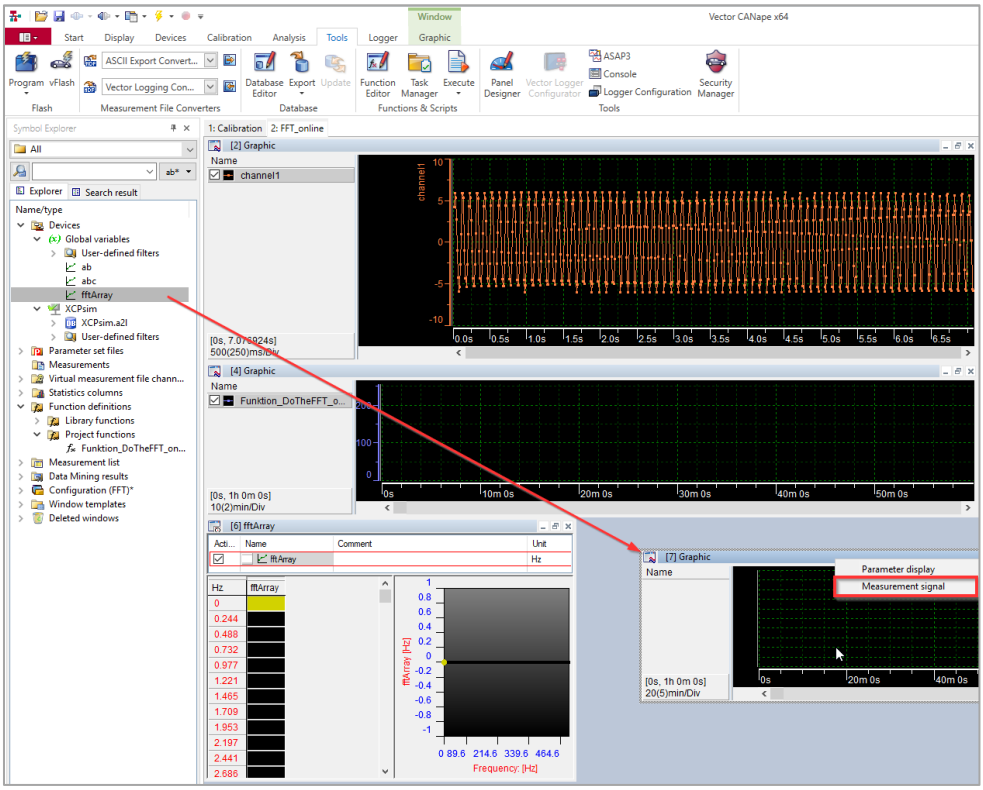
To visualize FFT spectra while computing the analysis, display the global variable 'fftArray' in the measurement window Map/Curve window. Just drag and drop the global variable from the symbol explorer to an empty space of your display page and follow the drop-down menu. While measuring Map/Curve window shows the result of a single FFT spectrum for the current time block. After the measurement was completed, the result of the last FFT spectrum will be displayed.

Note

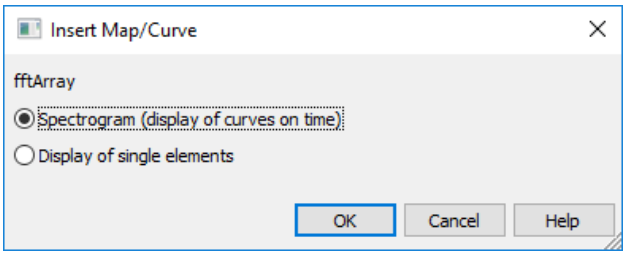
The calibration window MAP/Curve window will not display the result of the FFT while computing the analysis! However, the calibration window should be used for offline evaluation of the FFT.



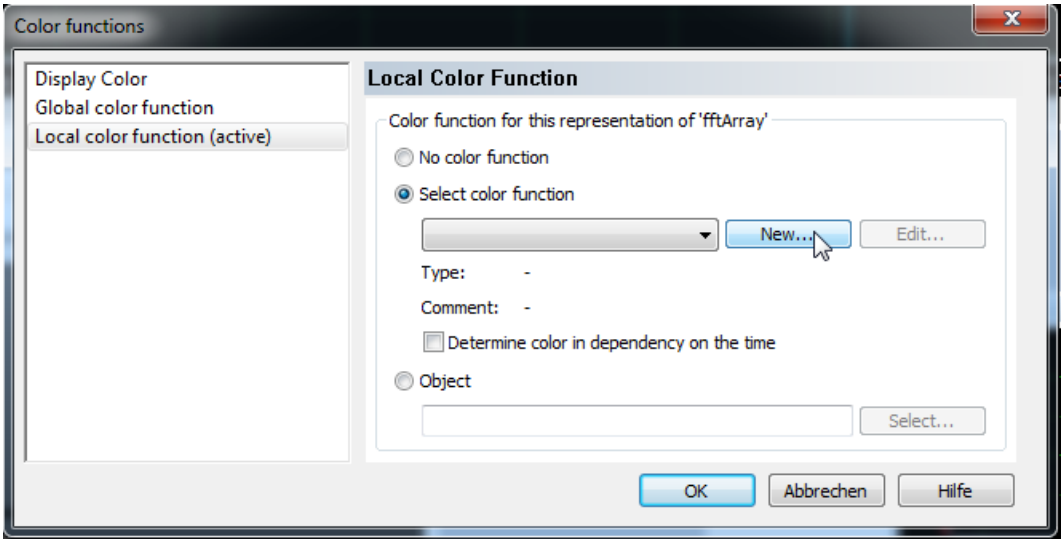
6. To display the FFT result in a spectrogram use the drag and drop functionality to display the global variable in a graphic window.



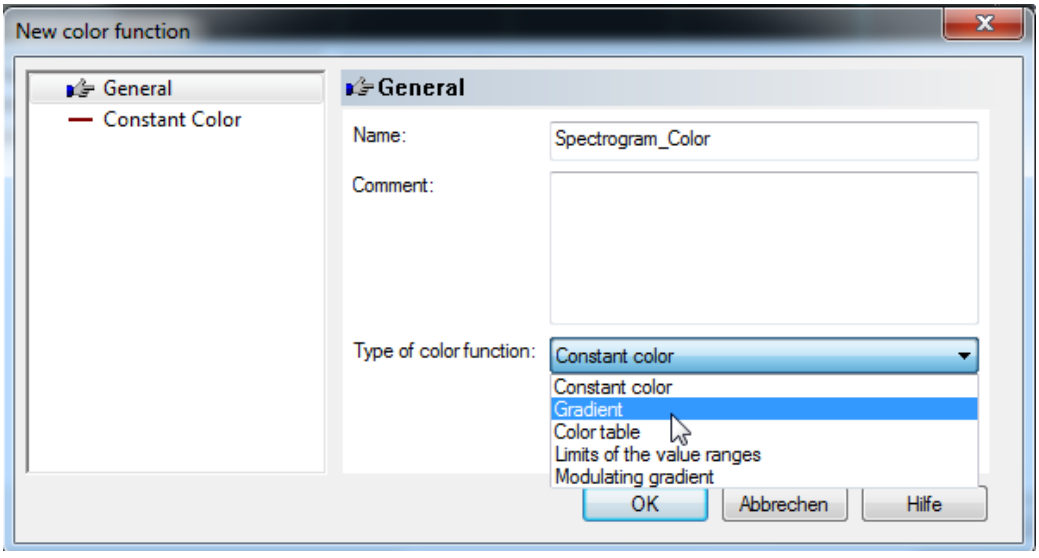
Insert the global variable as a measurement signal. A dialog opens asking to wither display the global variable as a spectrogram or just some of its elements. Select the option **Spectrogram**.



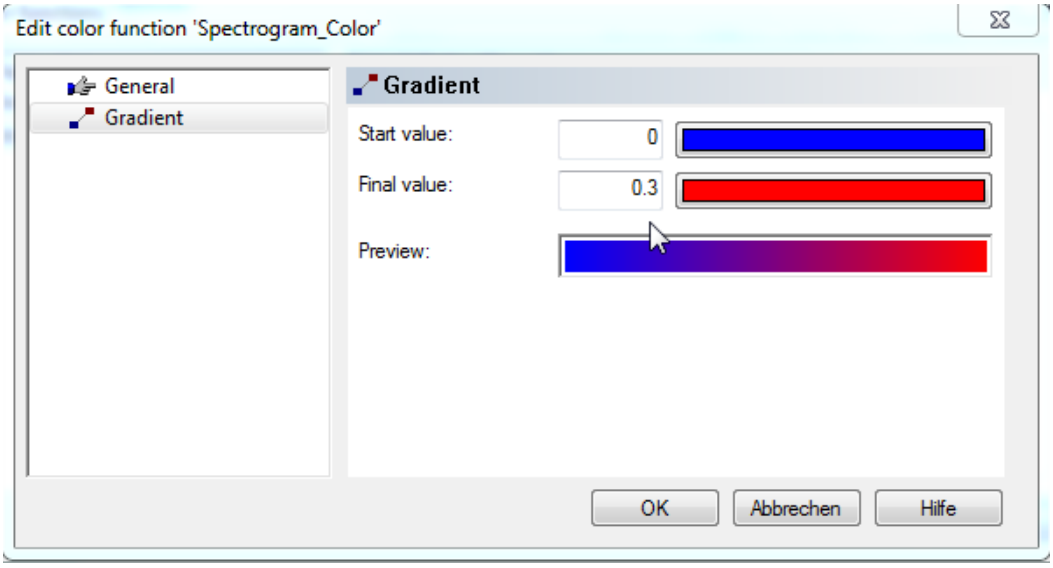
Choose a color function to display different amplitudes of the FFT spectra indifferent colors. E.g. choose the option **Local color function**, select **Select color function**, and click the button **New…**.



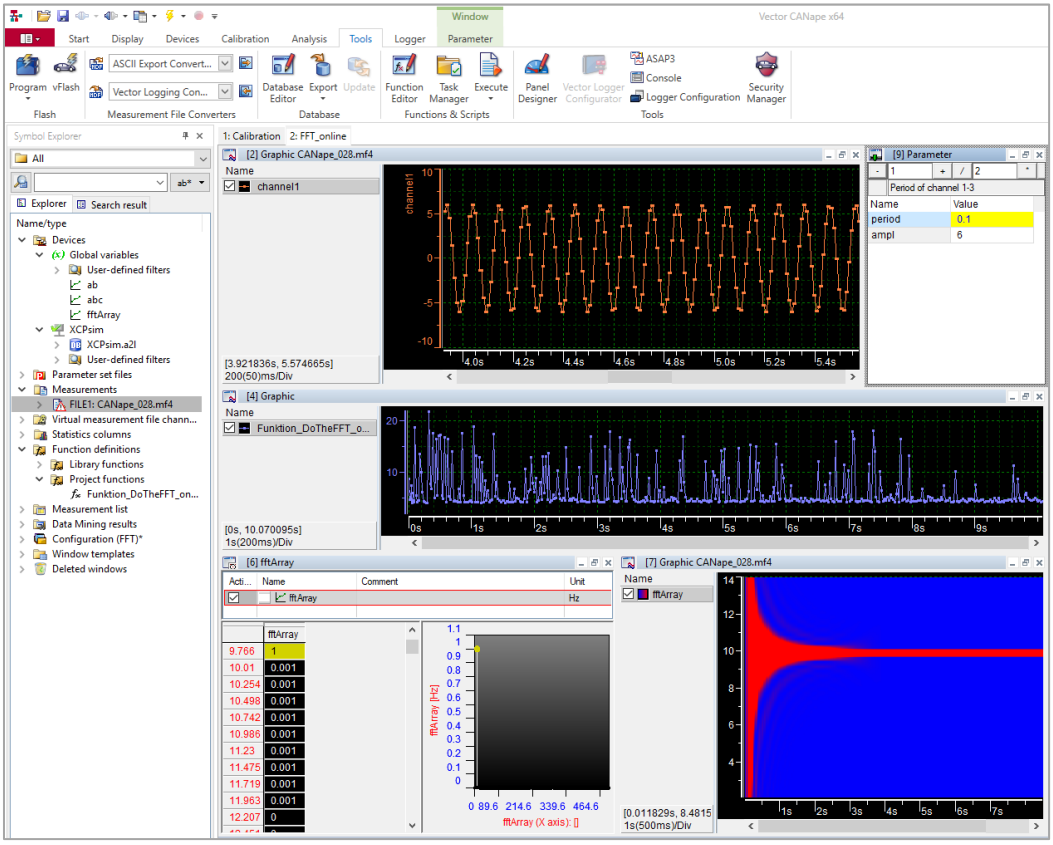
Name the new color function e.g. ‘Spectrogram_Color’ and change the function type to **Gradient**



In the function code above the option to normalize the FFT spectra is selected. Hence, the value range of the color function should be adapted. In this example the value range for the gradient is set from 0 to 0.3 and the color from blue to red for better visualization.



7. Start the measurement and observe the evaluation of the FFT in time.



Since the period of sine wave is 0.1s (see ‘period’-parameter in the Parameter window), the resonance at 10 Hz can be observed in the spectrogram.

1.5 Function code for offline evaluation of a FFT analysis

```
/*****************************************************************************
this function calculates the FFT over the previous BUFFER_SIZE number of
datapoints.
The current implementation is trimmed for offline analysis. The fft
analysis is only executed by the last recorded datapoint of the signal at
hand.

Within this function six parameters are defined (with R and I denote the
real and imaginary part of the FFT):

BUFFER_SIZE = defines the number of datapoints that are taken into account
by the function BlockFFT. If the number of datapoints previous to the
current analyzed datapoint is less than BUFFER_SIZE then the missing
parameter values for BlockFFT are omitted.

    BLOCK_FFT_FLAG_NORMALIZE = flag to normalize the result of BLOCK_FFT
    BLOCK_FFT_MODE_PEAK_AMPLITUDE = ( R^2 + I^2 )^0.5
    BLOCK_FFT_MODE_RMS_AMPLITUDE = ( ( R^2 +I^2 )^0.5 ) / ( 2 )^0.5
    BLOCK_FFT_MODE_AUTO_SPECTRUM = R^2 + I^2
    BLOCK_FFT_MODE_POWER_SPECTRUM = ( R^2 + I^2 ) / 2
*****************************************************************************
********/

function Funktion_DoTheFFT_offline (var signal, var fftArray[])
{
/********************** DEFINE PARAMETERS ******************************/
  #define BUFFER_SIZE   512
  #define BLOCK_FFT_FLAG_NORMALIZE                0x0001   // Flag,
can/must be combined with a mode
  #define BLOCK_FFT_MODE_PEAK_AMPLITUDE           0x0002
  #define BLOCK_FFT_MODE_RMS_AMPLITUDE            0x0004
  #define BLOCK_FFT_MODE_AUTO_SPECTRUM            0x0008
  #define BLOCK_FFT_MODE_POWER_SPECTRUM           0x0010

/************************ LOCAL VARIABLES ******************************/
  double buffer[BUFFER_SIZE];
  double n;
  double size;
  double t_end;
  double dim;
  double checkVal;
  long initial = 1;
  long idx;
  long isPowerOfTwo;


/************************ MAIN CODE ************************************/

  // ----------initialize variables----------------
  if (initial == 1)
  {
    initial = 0;

    // time of last measured datapoint
    size = sizeof(signal) - 1;
    t_end = time(signal[size]);

    // check if dimension of fftArry is less than 4096
    dim = xdimension(fftArray);
```

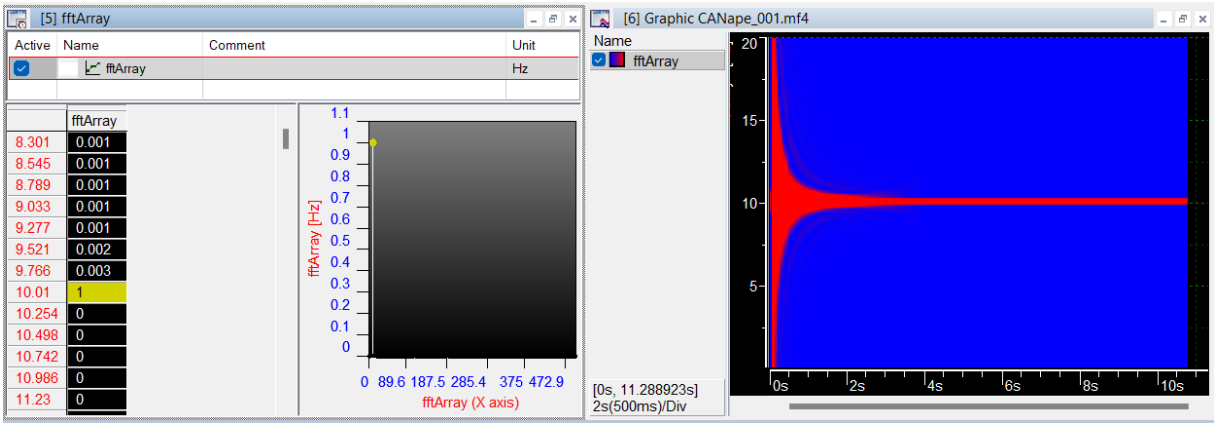
```
    if(dim >= 4096)
    {
     Write("The number of columns of fftArray is too large (currently %d).
The calculation cannot be computed and is stopped. Reduce the number of
columns to a value less than 4096. NOTE: The number of columns has to be a
power of two. ", dim);
        cancel;
    }

    // check if dimension of fftArray is a power of two
    isPowerOfTwo = 1;
    checkVal = dim;
    while((checkVal != 1) && (isPowerOfTwo == 1))
    {
      if(checkVal%2 == 0)
      {
        checkVal = checkVal / 2;
      }else{
        isPowerOfTwo = 0;
      }
    }
    if(isPowerOfTwo == 0)
    {
      Write("The number of columns of fftArray is not a power of two
(currently %d). The calculation cannot be computed and is stopped. Adjust
the number of columns.", dim);
      cancel;
    }

    // initialize global array fftArray, all elementens equal to zero
    for(idx = 0; idx < dim; idx++)
    {
      fftArray[idx] = 0;
    }
  }

  if (time(signal) >= t_end)
  {

    // copy the last BUFFER_SIZE number of datapoints to the array buffer
to be FFT analyzed.
    for (n = 0; n < BUFFER_SIZE; n++)
    {
      buffer[n] = signal[-n];
    }

    //calculate FFT
    BlockFFT(buffer, BUFFER_SIZE, BLOCK_FFT_FLAG_NORMALIZE |
BLOCK_FFT_MODE_POWER_SPECTRUM, fftArray);
  }
  return;
}
```

Note

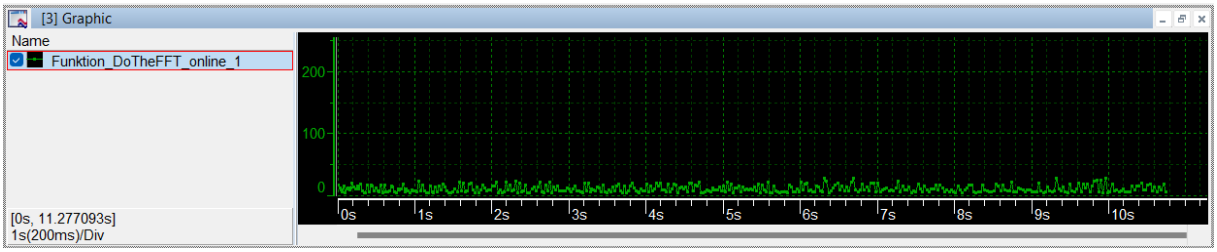
The display of spectrogram is not possible for the offline evaluation of a FFT analysis!

1.6 How to use the CANape demo project

The instructions shown in the previous subchapters are based on the CANape project provided with this Support Note. After loading the CANape project, the FFT analysis result of the sample measurement `CANape_001.mf4` is presented in the Map/Curve window [5] and in the Graphic Window [6].



In both graphs the dominant frequency can be observed at 10 Hz.
The user can start a new measurement connecting the XCP simulator which runs in the background. This would illustrate the 'online' use case explained in detail in the subchapter 1.4. After starting the measurement, the value of the `period` parameter is set to 0.1 to simulate the dominant frequency of 10 Hz. In addition, the calculation time of the `Funktion_DoTheFFT_online` function is displayed in the graphic window [3].



This graphic has no additional value for the FFT analysis itself, but can be used to monitor the calculation of the `Funktion_DoTheFFT_online` function.

1.7 Associated Files

SN-IMC-1-004_FFT-Analysis_CANape.zip: This file contains the two FFT-functions (`Funktion_DoTheFFT_online()` and `Funktion_DoTheFFT_offline()`) and a CANape demo project.

2 Contacts

For support related questions please address to the support contact for your country https://www.vector.com/int/en/company/contacts/support-contact/.