Exercises for imc FAMOS I – Digital Course

- Block 4 - Doc. Rev.: 1.2- 27.08.2025

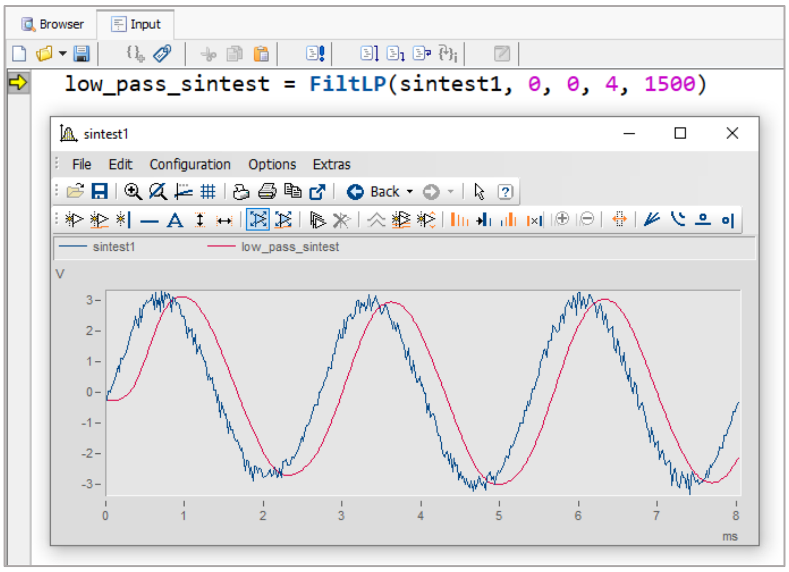
Exercise A

Exercise Objective:

Applying a function from the function library to a data set using the low-pass filter as an example.

Result:

The following image shows the line to be executed in the input area. The free floating curve window shows the original data set as well as the filtered data set.



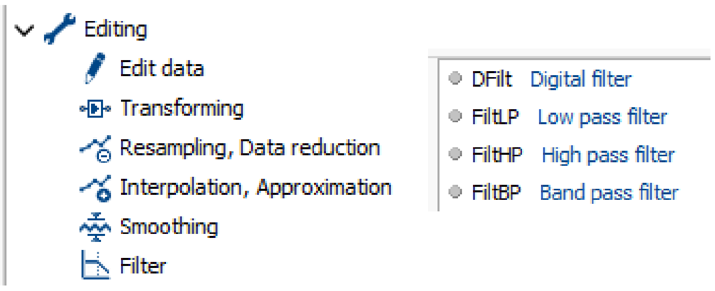
```
low_pass_sintest = FiltLP(sintest1, 0, 0, 4, 1500)
```

Exercise steps:

- Load the **sintest1.dat** dataset from the sample datasets.
- To execute functions, they must be entered with their correct syntax in the input window. Enter the following line directly in the input window using the keyboard and execute it using the "Ctrl+Return" key combination:

  low_pass_sintest = FiltLP(sintest1, 0, 0, 4, 1500)

  Alternatively, navigate to the **Editing** group in the function library and drag and drop the **sintest1** channel onto the **Low pass filter** function in the **Filter** subgroup.



- The input wizard then opens, with the help of which you can configure the parameters of the function as desired. Subsequently, the correspondingly configured line can be copied into the input window by clicking on **Copy** and executed as before. The execution can also be carried out directly in the input assistant by clicking on **Execute**.

For further study:

- Create another data set by applying the function **Smo5()**, which also suppresses the noise, twice to the channel **sintest1**:

  ```
  gla_sintest1 = Smo5(Smo5(sintest1))
  ```

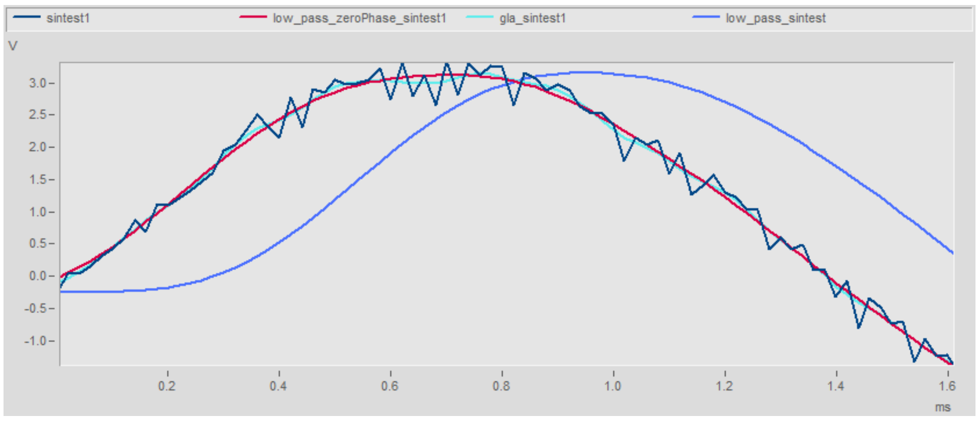
- Compare this with the result of the low-pass filter by displaying all data sets in a common curve window.

- In case you use a FAMOS license of the edition **Professional** or higher, ZeroPhase filters are additionally available. Create another filtered data set using the ZeroPhase lowpass filter.

  ```
  low_pass_zeroPhase_sintest1 = FiltLPZ(sintest1, 0, 0, 4, 1500)
  ```

- Then also compare this result with the previous ones (in terms of phase, residual noise, etc.).

  The result is expected to look something like this:



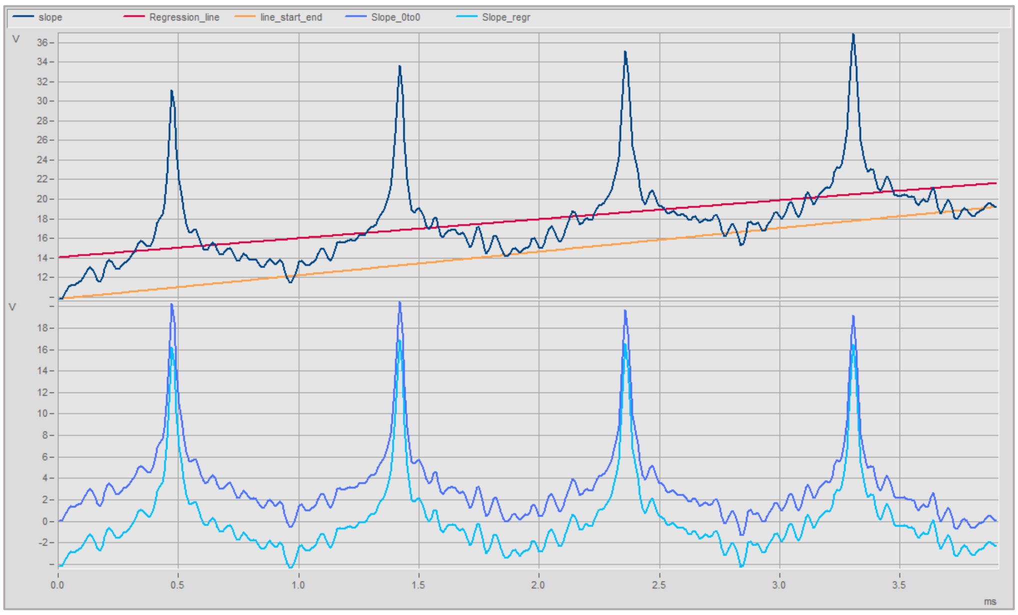
Exercise B

Exercise Objective:

A linear trend of a data set is to be corrected on a data set. This goal is realized in the exercise in two different ways. In the first case, the regression line of the data set is used. In the second case, the start and end points are simply used to determine a straight line for trend correction. The correction is done by subtracting these from the original data set in each case.

Result:

The following curve window shows the initial data set, the created auxiliary lines and the results after trend correction.



Exercise steps:

Method 1: Use of the regression lines

- Load the dataset **slope.dat** from the sample datasets.
- Create a regression line of **slope** using the function **Regr()**.
- To use the regression line for trend correction, it must have the same sample rate as the original data set. Use the **RSamp()** function to adjust the sample rate accordingly.
- Calculate the corrected data set by subtracting the regression line from the original data set.

Solution hint: The resulting code in the input window should look something like this:

```
Regression_Line = Regr(slope)
Regression_Line = RSamp(Regression_Line , slope)
Slope_regr = slope - Regression_Line
```

Method 2: Using the start and end points of the original data set.

- Load the dataset **slope.dat** from the sample datasets.
- Create a new auxiliary data set containing the first as well as the last y-value of the data set **slope**. Tip: The value of a data set at a specific index position can be accessed directly using **data[index]**.
- Create another auxiliary data set that contains the first as well as the last x-value of the data set **slope**. Tip: For the determination of the x-values the offset aswell as the length of the data set can be used.
- Using the two auxiliary data sets, now create a new data set that represents the correction line. To do this, use the **XYof()** function.
- To be able to subtract the correction data set from the **slope** data set, both data sets must have the same sampling rate. Therefore, perform a resampling of the correction line with the **RSamp()** function by using the **slope** data set as reference.
- Apply the trend correction by subtracting the correction data set from the **slope** data set.

Solution hint: The resulting code in the input window should look something like this:

```
y_values = [slope[1], slope[Leng?(slope)]]
x_values = [(xOff?(slope), xOff?(slope) + (Leng?(slope)-
1)*xDel?(slope)]
line_start_end = XYof(x_values, y_values)
line_start_end = RSamp(line_start_end , slope )
Slope_0to0 = slope - line_start_end
```

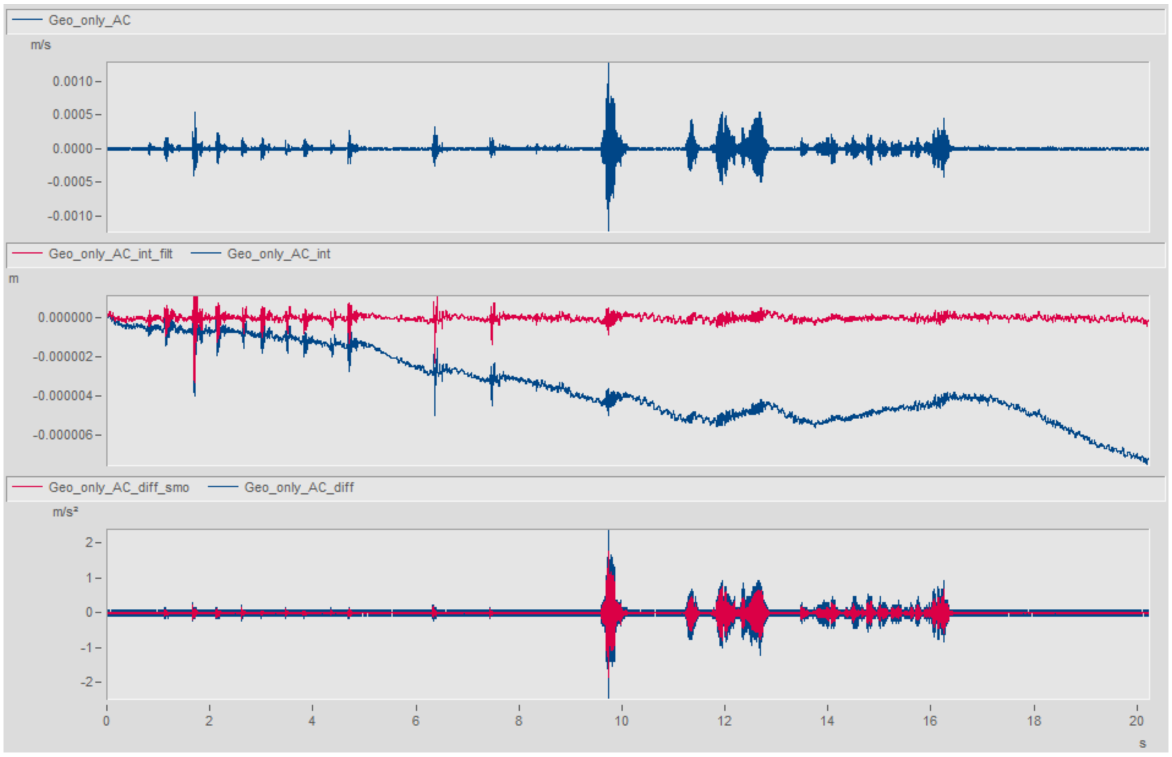
Exercise C

Exercise Objective:

This exercise deals with how to recognize and solve common problems in integration and derivation. To illustrate this, the vibration acceleration and the vibration displacement are calculated from a measured vibration velocity.

Result:

The following curve window shows the original data set aswell as the integrated and differentiated signals before and after their correction.



Exercise steps:

- Load the data set **AC_Data_Sample.dat** from the sample data sets and show it in a curve window. The signal was recorded using a geophone, which shows similar behavior to an IEPE sensor for acceleration measurement (piezoelectric sensor). These sensors do not record static or very slow changes.

Integration

- Integrate the data set using the **Int()** function from the **Basic Functions / Integration Differentiation** group and show the result in the curve window as well:

```
Geo_only_AC_int = Int(Geo_only_AC)
```

➔ An offset present on the data set is summed up by the integration and thus shows a drift in the integration.

- To suppress the erroneous summation of the offset, apply a high pass filter with a very low cutoff frequency (e.g. 1 Hz) either before or after integration:

```
Geo_only_AC_int_filt = FiltHpZ(Geo_only_AC_int, 0, 0, 2, 1)
```

- Add the filtered data set to the curve window as well. Vary the cutoff frequency of the high pass filter and pay attention to the changes in the result data set.

Differentiation (Derivation)

- Differentiate the data set **Geo_only_AC** with the function **Diff()** and add the result to the curve window again:

```
Geo_only_AC_diff = diff(Geo_only_AC)
```

➔ Any noise present on the data set is amplified by differentiation.

- To suppress the amplification of the noise, apply one of the smoothing filters (**Smo3()**, **Smo5()** or **Smo()**) either before or after differentiation:

```
Geo_only_AC_diff_smo = Smo(Geo_only_AC_diff,0.002)
```

- Add the smoothed result to the curve window. Vary the smoothing functions or the smoothing interval and observe the noise reduction in the result dataset.

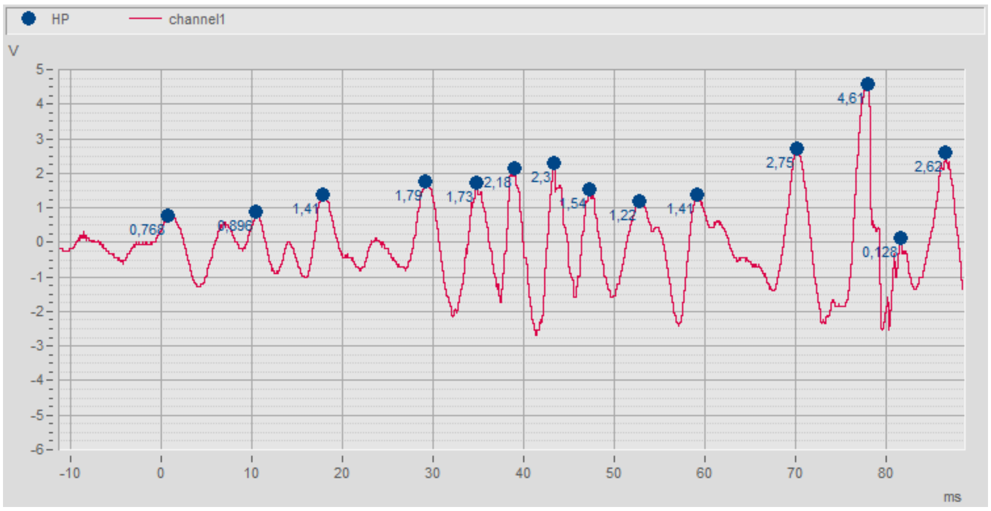
Exercise D

Exercise Objective:

From a data set, all relative maxima (high points) with a certain minimum amplitude between the extremes are to be determined. In the example used, only oscillations with a minimum amplitude of 1V are to be considered.

Result:

You will obtain the following result. All relative maxima are calculated, that follow a valley with the minimum amplitude of 1V.



Exercise steps:

- In the data browser, navigate to the **EXPERIM** folder in the sample data sets and load the **CHANNEL1.DAT** channel from the **0001** measurement folder. Then show the loaded channel in a curve window.
- To eliminate all oscillations smaller than the desired minimum amplitude, filter the data set with a hysteresis filter (**Hyst()** function) in an auxiliary data set.

```
help = hyst(channel1, 1)
hp = FindExtrema(channel1, "Local.Max")
```

- Open the properties of the curve window and switch to the **Lines** topic. Adjust the settings for the dataset of the maxima in a way that they are only shown as points with the corresponding label.

older versions prior to FAMOS 2025:

- Determine the x-values of all relative maxima using the **xMaxi()** function.

```
x_Koord = xMaxi(help, -1e35)
```

- The associated y-values can be determined with the **Value()** function.

```
y_Koord = Value(channel1 , x_Koord)
```

- Create a new xy data set from the x and y values that contains all the desired relative maxima. Then add this data set to the curve window.

```
HP = XYof(x_Koord , y_Koord)
```

- Open the properties of the curve window and switch to the **Lines** topic. Adjust the settings for the dataset of the maxima in a way that they are only shown as points with the corresponding label.