# Exercises for imc FAMOS II – Digital Course

**- Block 1 -**

Doc. Rev.: 1.2- 27.08.2025

# Exercise A

## Exercise objective:

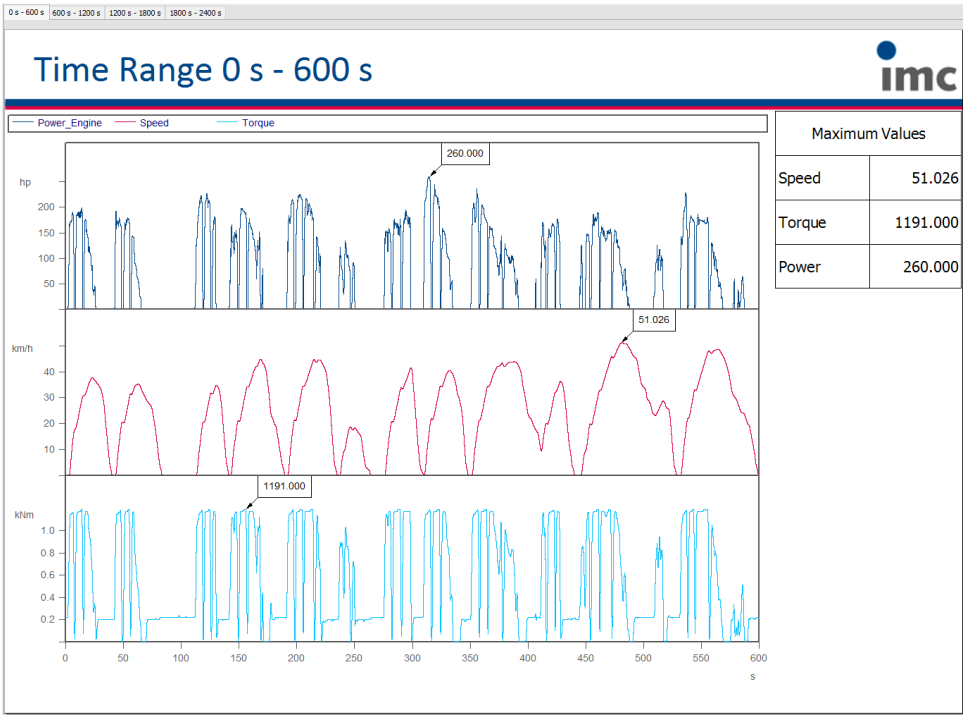
The goal is to develop a sequence that divides a longer data set into several sections of equal length and to create a separate panel page for each section. On the panel pages the respective section as well as some evaluated parameters of the individual sections are to be shown.

The techniques used for this are the function groups for using the curve window and the panel. In addition, commands for dealing with times, cutouts and loops will be introduced and deepened.

## Task formulation:

- Using the channels from the **bustrip.dat** sample dataset, create several panel pages by using the **Ex1.panel** template, which continuously display 600 seconds each of the measurements in the curve window (0-600 s, 600-1200 s, ... etc.).
- The measurement data are to be arranged on top of each other in the curve windows of the panel pages.
- Label the panel pages (tab labeling) as well as the heading text boxes in the panel pages with these range limits.
- Calculate the maximum values for each range and each channel and enter them in the respective tables as well as markers in the respective curve windows (results always with 3 decimal places).
- Create a report by exporting the panel pages as a PDF.

## Result:




The result panel should look like the adjacent graphic.

In addition, you will receive a PDF as a report with analog content.

## Exercise steps:

Preparation

- First load the panel **Ex1.panel**, which is included in the downloaded sample files. Then load the data set **bustrip.dat**, which is also in the downloaded sample files. After loading, the 3 channels **Speed**, **Torque** and **Power_Engine** should be present in the variable list.

Building the Curve Window Configuration

- Show the measurement data in a free-flying curve window (y-axes on top of each other).
- In order to be able to see the inserted markers later on, create some space above the respective curves by setting the **position top** to 80% for all y-axes in the curve window configuration (tab **Arrangement**).


- Save this curve window configuration as **bustrip.ccv** in the suggested default directory and close the curve window. The embedded curve window in the panel is to be loaded later with this configuration.
  Note: You can also find the **bustrip.ccv** file in the downloaded sample files.

Creating the loop

- Create a new sequence by clicking on the corresponding icon in the input window and save it.
- Using a **While** loop, the panel pages are now to be generated one after the other. To do this, the temporal positions of the left and right edges of the first time window are initially defined with two auxiliary variables. Each page is supposed to display a 600 second section of the **bustrip** measurements:

```
Range0 = 0; Left edge of the 600 s time window
Range1 = 600; Right edge of the 600 s time window
```

- The **While** loop is to run until the left margin **Range0** is less than or equal to the duration of a channel ( e.g. **Speed** in this example). At the end of each loop pass, both edges are incremented by 600 seconds to be used for the next loop pass.:

```
; as long as time window < channel duration
While Range0 <= XDel?(Speed)*(Leng?(Speed)-1)
   ; further code
   Range0 = Range0 + 600 ; Advance 600 seconds
   Range1 = Range1 + 600 ; for next loop pass
End
```

Creation of the loop content

- The functional part of the loop is developed within the loop in the following. Make sure that the updates of **Range0** and **Range1** always remain last at the end of the the cycle.
- The individual panel pages are created by copying the first panel page at each loop pass. For the naming of the panel pages the scheme "0 s - 600 s" (as an example for the first block) is to be used.

```
Paneltext = TForm(Range0, "f1.0")+" s - "+TForm(Range1,"f1.0")+" s"
PnInsertPage("", 1, Paneltext, 0)
```

- Find out the name of the embedded curve window in its widget properties and load the curve window configuration you created earlier.

```
CwLoadCCV("Curve1", "bustrip.ccv")
```

- Next, the range limits for the x-axis are to be set. To do this, first select the curve window and then the contained x-axis:

```
CwSelectWindow("Curve1")
CwSelectByIndex("x-axis", 1)
```

- Then set the range limits of the X-axis using the range variables. Before this, however, the range selection must always be set to "fixed" to ensure that these are adopted.:

```
CwAxisSet("range", 4); always set Range to "fixed limits" first
CwAxisSet("min", Range0)
CwAxisSet("max", Range1)
```

- In order to calculate the maximum values for each section, the respective sections to be displayed must be cut out of the data. Afterwards, the maxima of the sections can be determined:

```
CutSpeed = Cut(Speed, Range0, Range1)
CutTorque = Cut(Torque, Range0, Range1)
CutPower = Cut(Power_Engine, Range0, Range1)
MaxSpeed = Max(CutSpeed)
MaxTorque = Max(CutTorque)
MaxPower = Max(CutPower)
```

- Put the calculated section maxima into the table on the panel. Again, first find out the name from the widget properties:

```
PnTableSetCell("Tab1", 2, 2, TForm(MaxSpeed, "f1.3"))
PnTableSetCell("Tab1", 2, 3, TForm(MaxTorque, "f1.3"))
PnTableSetCell("Tab1", 2, 4, TForm(MaxPower, "f1.3"))
```

- The header of the panel page is defined by a label widget. Set the heading such that it incorporates the respective Time Range:

```
PnSetText("LB_Start", "Time Range " + Paneltext)
```

- Next, markers are to be placed at the maximum positions of all curves. Since there is more than 1 coordinate system in the curve window configuration, the curve line of the relevant channel must first be selected. Start with the **Speed** channel:

```
CwSelectByChannel("Line", Speed) ; Selects the line of the channel
```

- Next, create a marker and assign it to the selected line:

```
CwNewElement("marker")
CwMarkerSet("line.selected",1); Placement of marker on selected line
```

- Finally, the marker must be positioned correctly, coordinates should be interpreted as a physical unit (and not as % of the axis length):

```
CwMarkerSet("x.type", 1); Physical units are used
CwMarkerSet("y.type", 1); see above
CwMarkerSet("x", Pos(CutSpeed, MaxSpeed)); Position X (Max from
section)
CwMarkerSet("y", MaxSpeed); Position Y
CwMarkerSet("text", TForm(MaxSpeed, "f1.3")); Markertext
```

- Repeat all marker creation steps for **Torque** and **Power_Engine** channels. Make sure to select the correct data set first.
- After the loop has been completely processed, the first page, which served as a template, is to be removed. Then export the panel as a PDF:

```
PnRemovePage(1); get rid of empty dialog page
PnExportPDF("C:\temp\Report.pdf", 0, 0)
```

- Save the resulting sequence.

# Exercise B

## Exercise objective:

The aim of the exercise is to apply an evaluation of a data set, in which certain characteristic values are determined, to a whole set of data sets in parallel. The techniques used include cutting data sets, working with groups and loops, and generating data sets with fixed values from scratch..

## Task formulation:

Determine the mean values of the individual plateaus from the sample data set **LoadTestSample.dat** for the variables **load1..4**.

To accomplish the exercise, follow these steps:

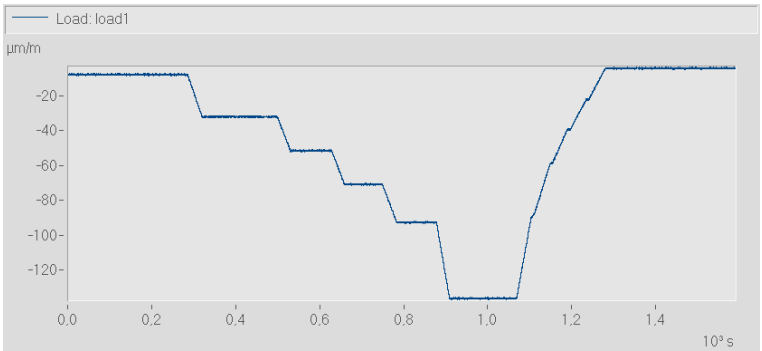
- Using the measurement cursors, determine approximately the time midpoints of the plateaus and create a channel that includes all these values.
- First, use a loop to cut out a 20-second piece of data for the first of the 4 data sets at all determined times on a test basis..
- Determine the mean values of each of these pieces in a new data set that includes the respective temporal midpoint used in its name.
- To perform the calculation simultaneously for all **load1..4**, group these variables afterwards and perform the generated sequence with the group itself instead.

## Result:

For each of the 7 plateaus from the data sets **load1..4** you get a new group, which contain the mean values of all 4 data sets for the respective plateaus. The names of the groups contain the temporal midpoints of the plateaus.

## Exercise steps:

- Load the downloaded dataset LoadTestSample.dat via the FAMOS browser into the variable list. Afterwards the variables **Force1..2**, **load1..4**, **displacement1..3** should appear in the variable list.
- Show the dataset **load1** in a curve window. Here you can see a total of 7 plateaus with approximately the same load, each of which is present for a different duration.



- Use the measurement cursors to determine the approximate time centers of the plateaus and note them down. Create a **Time_Of_Interest** channel containing the noted times and assign it the y-unit "s":

```
Time_Of_Interest = [140, 410, 580, 700, 830, 990, 1440]
SetUnit(Time_Of_Interest, "s", 1)
```

- Let's first create the loop structure over the 7 plateaus:

```
FOR i = 1 TO 7
   ; Loop code
END
```

- At the beginning of each loop pass, the start and end time of the piece of data to be cut out is to be determined for the i-th plateau. Since the length of the data to be cut out is to be 20 seconds in each case, the time points sought can be defined as follows:

```
x1 = Time_Of_Interest[i] - 10 ; Start time
x2 = Time_Of_Interest[i] + 10 ; End Time
```

- Cut out the corresponding piece of data:

```
CutLoad = Cut(load1, x1, x2)
```

- Create a new text variable consisting of the name **MeanLoad_** followed by the respective entries from the **Time_of_Interest** dataset.

```
TxMeanLoad = "MeanLoad_" + TForm(Time_Of_Interest[i],"")
```

- Use the created text variable to create a new variable with the name of the text containing the average value from the cut piece of data.

```
<TxMeanLoad> = Mean(CutLoad)
```

- Save the sequence and run it completely as a test run. You should then have 7 variables with the corresponding times in their names in the variable list, indicating the respective mean values of the plateaus.

Application to a group

- In the following, the calculations of the sequence are to be carried out for all **load** variables simultaneously. To do this, select **load1** to **load4** in the variable list and select **Group…** in the context menu (right-click) to combine the selected variables in a group. Enter **load** (without a number) as the group name.
- In order to apply the created sequence to the group, the variable **load1** must be replaced by the group name **load** in the sequence.
- Execute the sequence. The result is a new group for each time point of the **Time_of_Interest** data set containing the mean values of the individual cut plateaus from **load1..4**.
- Save the sequence.

The overall sequence now looks like the following:

```
Time_Of_Interest = [140, 410, 580, 700, 830, 990, 1440]
SetUnit(Time_Of_Interest, "s", 1)

FOR i = 1 TO 7
   x1 = Time_Of_Interest[i] - 10 's'
   x2 = Time_Of_Interest[i] + 10 's'
   CutLoad = Cut(Load, x1, x2)
   TxMeanLoad = "MeanLoad_" + TForm(Time_Of_Interest[i],"")
   <TxMeanLoad> = Mean(CutLoad)
END
```

# Exercise C

## Exercise objective:

This exercise is an extension of exercise B. Here, too, the aim of the exercise is to apply an evaluation of a data set, in which certain characteristic values are determined, to a whole set of data sets in parallel. In this case, the characteristic values are not simply output, but used directly for further evaluations. The techniques used here are linking data sets, working with XY data sets and sorting values.

## Task formulation:

Load the **LoadTestSample.dat** dataset and create the characteristic curves **displacement1** over **Force1** and **displacement2** over **Force2** from the mean values of their 7 plateaus. Show both characteristic curves in the same coordinate system and on the same axis. Use groups instead of single channels in the calculations from the beginning.

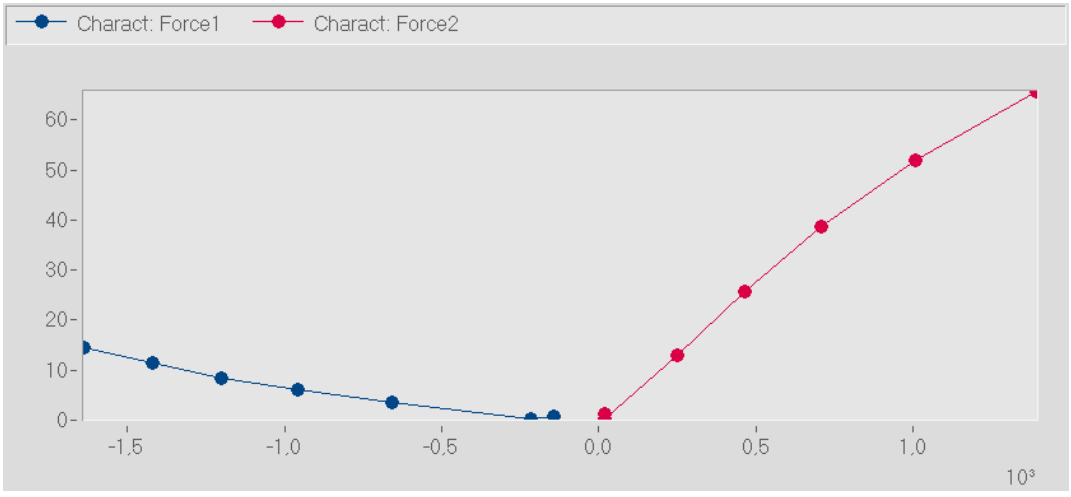
Note: The positions of all plateaus in **LoadSampleData.dat** match in time.

To perform the exercise, proceed as follows:

- Copy your source code from exercise B.
- In the variable list, group **Force1..2** to **Force** and **displacement1..2** to **Displacement**
- Determine the pieces **CutDisp** and **CutForce** analogously to **CutLoad** from exercise B.
- Create two datasets containing all mean values from **Displacement** and from **Force**.
- After the loop, combine the 2 data sets into a new XY data set **Displacement** over **Force** and show the characteristic curves in the curve window
- Sort the XY data sets according to the X values of the characteristic curves

## Result:

The result of this exercise provides a calculated characteristic curve for each of the data sets **Displacement** over **Force**, that are shown in the following curve window:



## Exercise steps:

- The calculations from exercise B are also to be applied to the **Force1..2** and **displacement1..2** channels. Instead of simply outputting the mean values of the individual plateaus in separate data sets, however, in this case the individual plateau mean values are to be used to create a **displacement**-**force** characteristic curve.
- Copy the contents of the sequence from exercise B into a new sequence.
- Group the channels **Force1..2** in a group **Force** as well as **displacement1..2** in a group **Displacement**.
- Add appropriate commands to the loop contents of the sequence to create sections for **Displacement** and **Force** analogous to the **load** sections:

```
CutDisp = Cut(Displacement, x1, x2)
CutForce = Cut(Force, x1, x2)
```

- Again, calculate the mean values for each plateau. However, instead of writing these to new variables, add the calculated values to a **DispTrd** and a **ForceTrd** variable so that they contain all the mean values of the plateaus at the end of the loop:

```
DispTrd = Join(DispTrd, Mean(CutDisp))
ForceTrd = Join(ForceTrd, Mean(CutForce))
```

- Since **DispTrd** and **ForceTrd** do not yet exist at the beginning of the loop, initialize them before the loop as empty variables:

```
DispTrd = empty
ForceTrd = empty
```

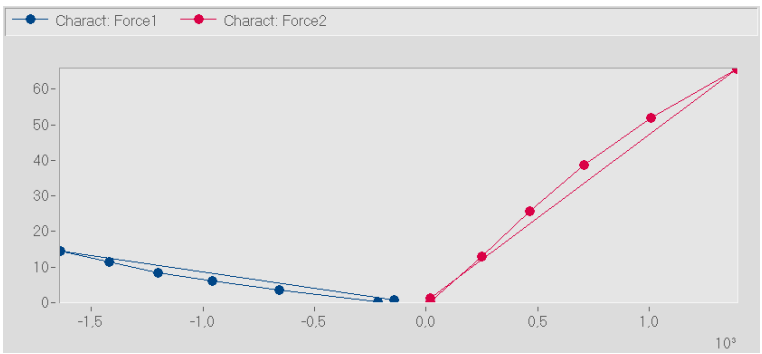
- After the loop, the characteristic curves are to be generated. For this purpose, combine the groups **DispTrd** and **ForceTrd** to an XY data set called **Charact**:

```
Charact = XYof(ForceTrd, DispTrd)
```

**DipsTrd** is set here as the Y coordinate and **ForceTrd** as the X coordinate. Again, FAMOS automatically recognizes that **ForceTrd** and **DispTrd** are groups and automatically forms a new group **Charact**.

**Caution!**
*FAMOS does not generate a warning here, although the same number of records are not present in the groups. Thus, the third channel **displacement3** is omitted here without the user noticing anything.*

- In front of the Charact group in the variable list, you can see the hysteresis symbol , because the X-components are not monotonically increasing. You can understand this by looking at the

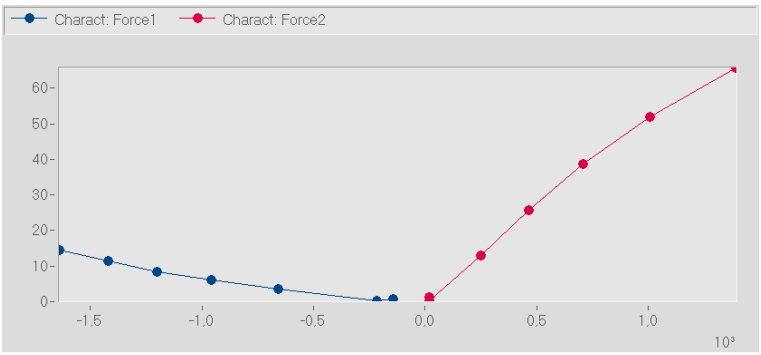


plateau curve of the **Force1** measurement, for example. The characteristic curve looks accordingly if you visualize **Charact** (1 coordinate system, 1 axis, lines with symbols):

- In order to present the characteristic curve as usual, sort the values of the characteristic curve group in ascending order according to the X-values:

```
Charact = Sort(Charact, 7)
```

When this command is executed, the variable symbols change to and you receive the following characteristic curve for the **Force** characteristic over **Displacement**:




- Save the sequence.

The full code looks as follows:

```
Time_Of_Interest = [140, 410, 580, 700, 830, 990, 1440]
SetUnit(Time_Of_Interest, "s", 1)
DispTrd = empty
ForceTrd = empty
FOR i = 1 TO 7
   x1 = Time_Of_Interest[i] - 10 's'
   x2 = Time_Of_Interest[i] + 10 's'
   CutLoad = Cut(Load, x1, x2)
   TxMeanLoad = "MeanLoad_" + TForm(Time_Of_Interest[i],"")
   <TxMeanLoad> = Mean(CutLoad)
   CutDisp = Cut(Displacement, x1, x2)
   CutForce = Cut(Force, x1, x2)
   DispTrd = Join(DispTrd, Mean(CutDisp))
   ForceTrd = Join(ForceTrd, Mean(CutForce))
END

Charact = XYof(ForceTrd, DispTrd)
Charact = Sort(Charact, 7)
```