**C++ Material Calibration tool**

This calibration tool is a continuation of the work developed with the Fortran program “Yeoh\_Prony.exe”. The program had some limitations in terms of available Hyperelastic models, Yeoh or Neo-Hookean, and it was only possible to perform the integrated hyper-viscoelastic calibration for Uniaxial and Equibiaxial test data.

Further work was made to extend the capabilities. The C++ program supports new Hyperelastic models: Ogden (order N=1 to N=6), Arruda-Boyce, Van Der Waals and Mooney-Rivlin. It also allows integrated hyper-viscoelastic calibration for the Planar test data.

**Note** (07/22/2016): Until this date only the Hyper-viscoelastic calibration for Neo, Yeoh and Ogden were validated.

****Folder Content****

To work properly, the user needs to have this files in the same folder:

* The material model input file “Hyper\_Prony\_Material\_Model.txt”
* The file “Data\_Suite.txt” declaring the test data files
* The files with the test data
* The executable “Hyper\_Visco\_Calibration.exe”

****Software workflow****

Figure 1- Software Workflow



**Hyper\_Prony\_Material\_Model.txt**



**Data\_Suite.txt**



**C++**



**[Filename]\_sim.txt**



**Data\_suite\_error.txt**

**Output Files**

**Input Files**

****Input File: “Hyper\_Prony\_Material\_Model.txt”****

This file allows the user to define a set of parameters to configure the Hyperelastic model and the viscoelastic Prony series. When executing, the C++ program parses the content of the file to gather relevant information like the defined Hyperelastic model and its parameters, and the number of Prony terms and its values.

Depending on the chosen Hyperelastic model, the data lines with the model parameters have to change accordingly. The parameters represented on the data line (figure 2) follow the Abaqus input file format (see Abaqus Keywords Reference Guide, section \*Hyperelastic).

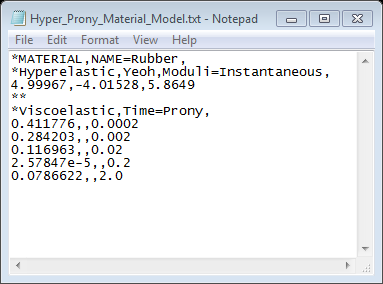
Note that each data line can only contain a maximum of 8 parameters. In cases where the user needs to exceed this amount a new data line must be created.

In terms of Prony series, only gs and τs are consider by the C++ program. However, for the ks values, the user must leave an empty space between commas, as shown in figure 2. It’s possible to define an unlimited number of Prony terms.

For user convenience, a folder “Input Files” is available with the template for each Hyperelastic model. It’s important to keep the input files format for the correct behavior of the C++ program.

Prony gs

Hyperelastic Model



Prony

Model parameters

Figure 2- Yeoh model input File

In figure 2 is represented the input file for the Yeoh model and the respective location of the parameters that the user can define and change accordingly to the material model. In some models the user needs to specify the order of the model using the parameter “N”. In figure 3 is an example of an Ogden Model input file with order N=2.

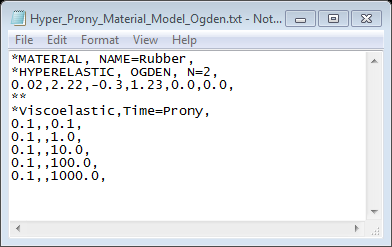


Figure 3- Ogden model input file example

The follow table shows how to define each model in the input file for each Hyperelastic model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hyperelastic Model Name | Model Definition | N Parameter | Hyperelastic Parameters | |
| Neo Hooke | Neo | NA | c10 | |
| Yeoh | Yeoh | NA | c10,c20,c30 | |
| Ogden | OGDEN | N=1 to N=6 | µi,αi,  (i=1 to N) | Di  (i=1 to N) |
| Van Der Waals | VAN DER WAALS | NA | µ,λm,α,β,D | |
| Arruda-Boyce | ARRUDA-BOYCE | NA | µ,λm,D | |
| Mooney-Rivlin | POLYNOMIAL | N=1 | C10,c01,D | |

Note that it’s possible to define a Neo-Hooke model using the Yeoh model template and setting the parameters C20 and C30 to “0.0”. Since the material is considered incompressible, the Ds values are set to “0.0”.

Input File: “Data\_Suite.txt”

This file contains the name of the files with the test data that the C++ program will use in the material model calibration.

The names of the files inside “Data\_Suite.txt” must be consistent with the names of the experimental data files. The name has to follow a specific format so that the program can distinguish between the types of experimental data available: uniaxial, equibiaxial or planar.

For each case, the file name format must be:

* Uniaxial: “UN\_[filename].txt”
* Equibiaxial: “EB\_[filename].txt”
* Planar: ”PT\_[filename].txt”

In figure 4 is an example of one “Data\_Suite.txt” file:

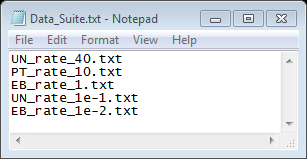


Figure 4- Data Suite file example

Test Data Files

The files with experimental data must be in the same folder as the C++ executable. The content of the files must be consistent in terms of format. In the beginning of the file, the three first lines are considered only for the header. The experimental data must start at the fourth line and have three columns of data in the following order: Time, Engineering Strain and Engineering Stress.

During the performed tests with Isight, It was noticed that better fitting results are obtained when the different files have the same amount of points. The user must manipulate the test data in order to obtain this.

Tip: To create this files easily, first insert the data into an excel spreadsheet. Format the numbers to have the same amount of digits. Then copy the data from Excel directly to a new Notepad. The file must look like the follow example:

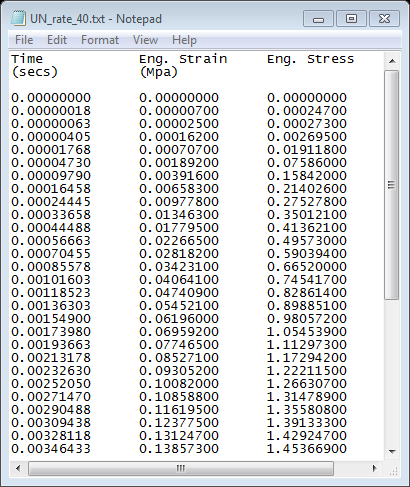


Figure 5- Test data file

Output Files

After each C++ program run, two types of text files are generated:

The first type are the simulation files with the results of the performed calibration. The program name them “[filename]\_sim.txt”, where [filename] represents the name of each experimental data set used. Compared to the original file, the simulation files have two more columns attached: “Calc Stress” and “Error”. The “Calc Stress” column represents the stress calculation per-point using the defined material model while the “Error” represents the error measure of each approximation.

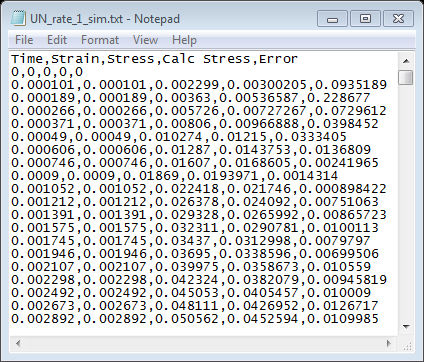


Figure 6- Output Simulation file

The second type is a file named “Data\_Suite\_Error.txt”. The content of this file represents the sum of the errors of each point for all the experimental data sets. In terms of the optimization the error values will be defined as objectives to be minimized.

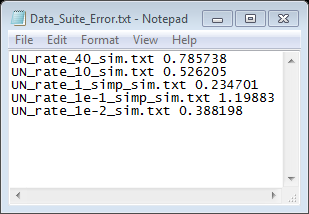


Figure 7- Output Error file

Error Measure

The approximation error between the simulation and the test data is calculated per point using the Relative Square Error formula:

The final results of the error measure are outputted to a new file called “Data\_suite\_error.txt” which contains the error measures per file, resulting of the sum of all error measures per point, normalized by the total number of points:

C++ Program

The C++ program is intended to be used inside an iSight optimization loop. In each run the program performs the Hyper-Viscoelastic calculations for the defined material model, calculating the associated error measure and storing the results in the output files.

To run the program the user must ensure that all the relevant files are in the same folder.

After defining correctly all the input files, the user should run the executable using the command prompt before starting the iSight configuration. This way it’s possible to check if the program runs successfully and creates all the necessary files to configure the optimization model.

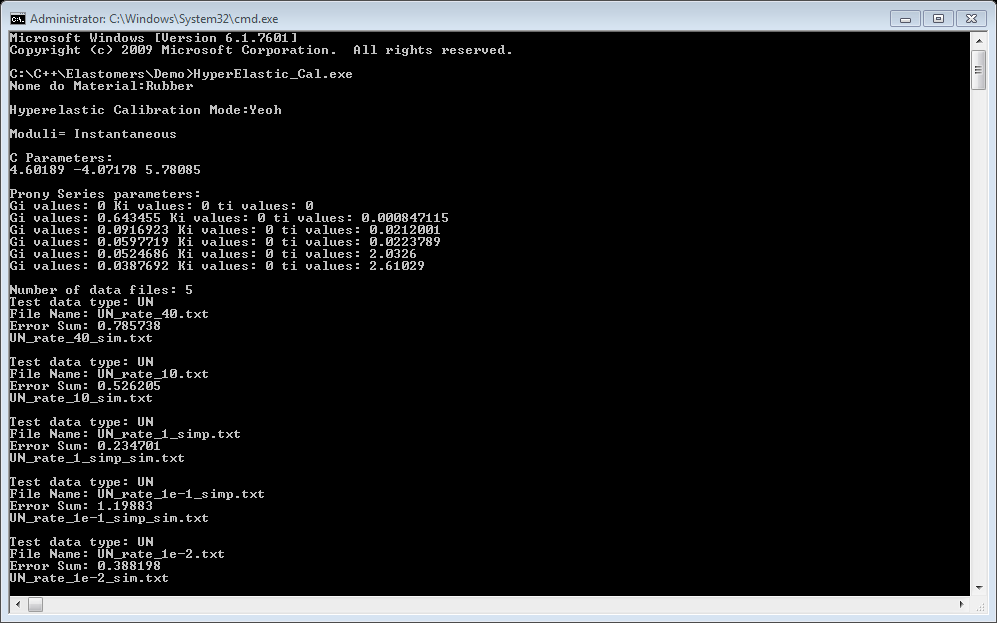


Figure 8- Single execution of the C++ program

Code Structure

**Implementations:**

-Read & Write functions

-Parse files functions

**Implementations:**

- Hyperelastic models  
- Viscoelastic calibration

- Error measure calculations

main.cpp

hyper\_Visco.cpp

hyper\_Visco.h

files\_IO.cpp

files\_IO.h

Compiler info

**Name**: GNU GCC Compiler

**Version**: 5.3.0

**Download** **page**: https://nuwen.net/

**Recommendation**: Create a folder on the main drive to install the compiler, e.g. “C:\MINGW”.

**IDE:** Code Blocks 16.01 (see appendix for the compiler configurations)

**Graphics GUI**

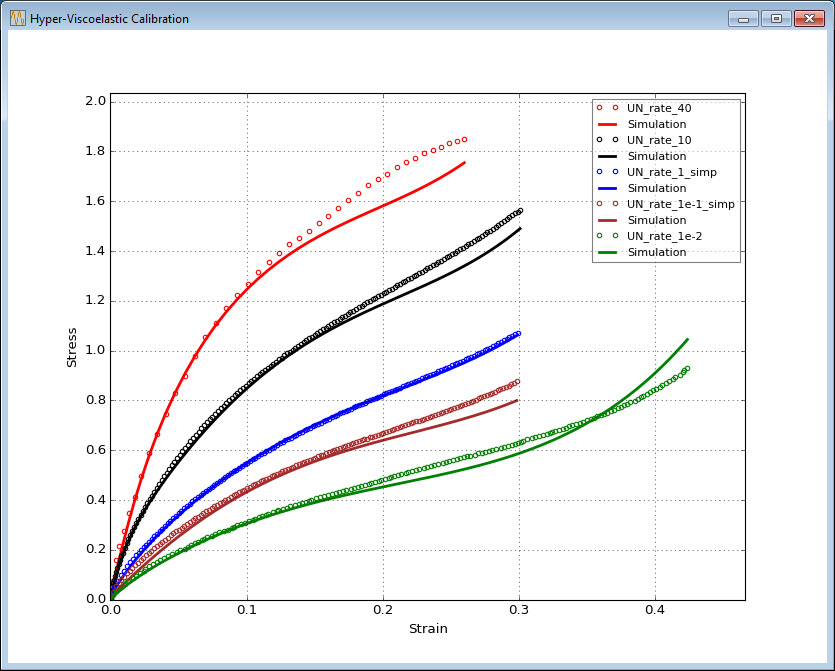
****

Figure 9- Graphic GUI to display the curve fitting

The GUI “Graphics\_GUI.exe” is a standalone application that allows the user to see the curve fitting process. It can be used with Isight as a complement to display the real time calibration. To function properly it needs the “Data\_Suite.txt”, the test data files and the generated Output files mention above. The maximum number of different plots allowed is 12.

The executable must be copied to the same working directory of the C++ program/Isight model.

This GUI was developed using Python version 2.7.3 with the external library Matplotlib. The compilation of the file to an executable was made using “PyInstaller 3.2”. The program was compiled statically to avoid dependency problems.

Isight Integration

Configure Isight project

When creating a new Isight model, it is necessary to configure some options in order to the workflow executes successfully. Following this configuration makes the model easily shareable, not relying on absolute paths.

Under the **Edit Menu** select “**Model Properties**”, then select the tab “**Execution**”. On this tab set the “Model runtime directory” to “{modeldir}” and uncheck the box “Create sub-directories”. Save the new model on the directory with all the files for the calibration process.

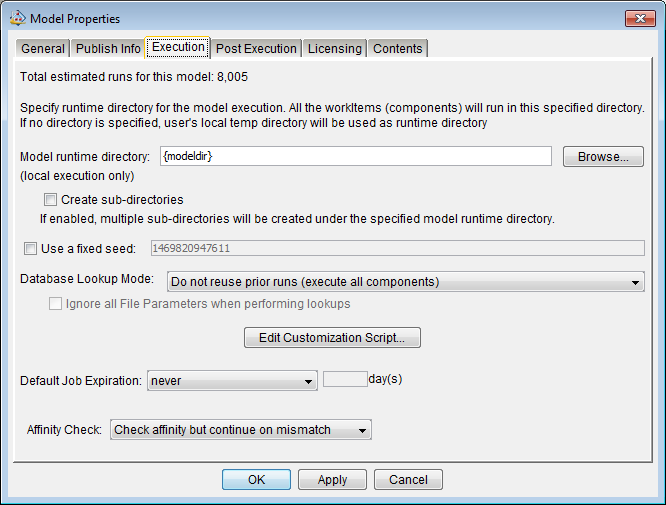


Figure 10- Isight model properties configuration

Isight Workflow

To perform the material model calibration different Isight workflows and optimization methods were tested. In figure 11 is represented the workflow from which was possible to obtain the best results in less time.

The workflow is constituted with an Exploration component, a Calculator, a Data Exchanger for the input file “Hyper\_Prony\_Material\_Model.txt”, an OS command component for the C++ program and another Data Exchanger for the error file.

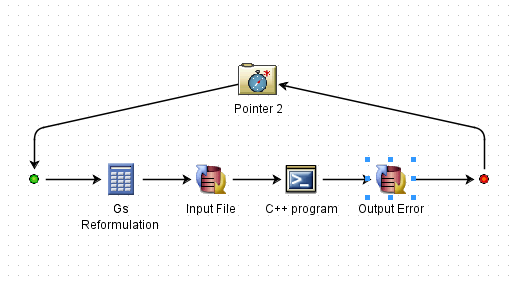


Figure 11- Isight Workflow

Input file Data Exchanger

In order to set the optimization parameters and update the results to the input file “Hyper\_Prony\_Material\_Model.txt” during the optimization a Data Exchanger component is necessary:

For a Hyper-viscoeleasticYeoh model the follow parameters must be configured as write variables:

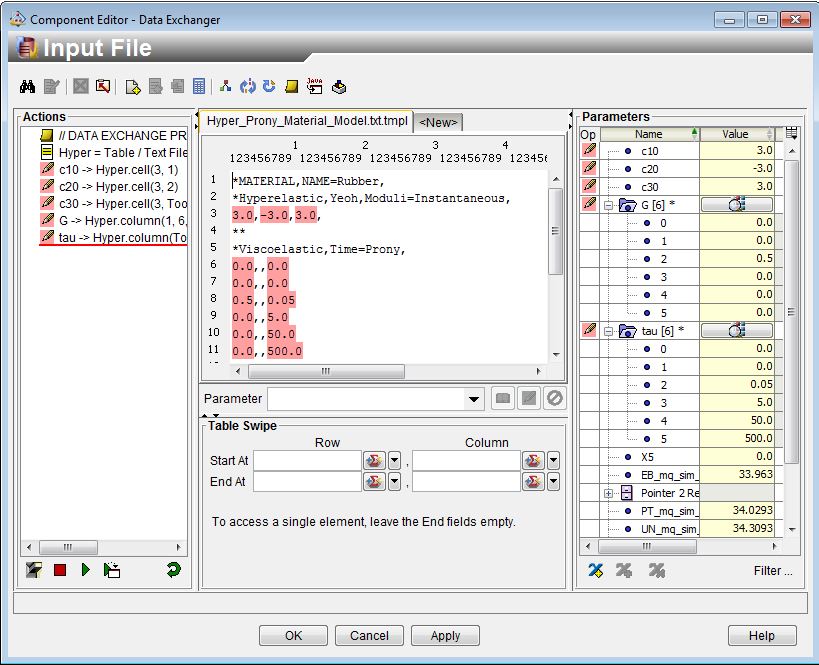


Figure 12- Configure Input Data Exchanger

Note that this input file has 6 Prony terms configured. Even if it is more than necessary, this is a good practice because it gives more flexibility to the user. This way it’s possible to test the model varying the number of prony terms without the need to reset the Data Exchanger. The number of Prony terms is then adjusted inside the Exploration component selecting the desired amount of Prony terms as variables.

After configure the data exchanger, go back to the Design Gateaway and, with the focus on the Input file component, select the “Files” tab. On this tab configure the Location to”File” and the path to {modeldir}.

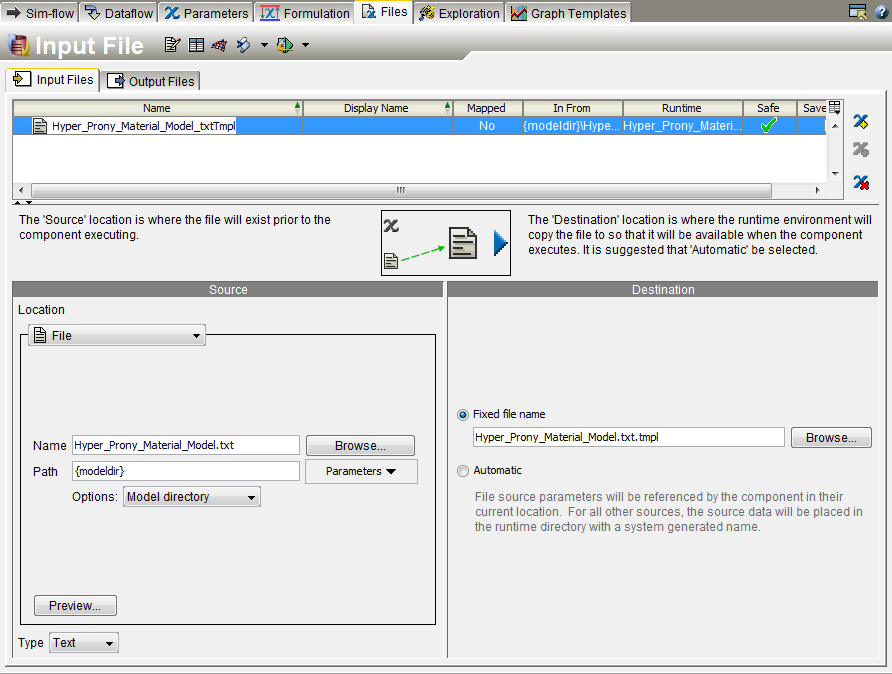


Figure 13- File properties configuration

Gs Reformulation Calculator

The Calculator component is used to constrain the values of the gs prony terms during the optimization, forcing. Considering the typical shape of the relaxation curve for rubber materials, this tends to be the regular behavior of the g values in a Prony series.

The reformulation  is defined as follows:



To implement it, e.g. using a five Prony terms model,the user must create 4 variables X2, X3, X4, X5. With the focus on the calculator component, select the “Parameters**”** tab. Then, press the button “New…” button, name the variable, and press “OK”. Repeat this procedure for the 4 variables.

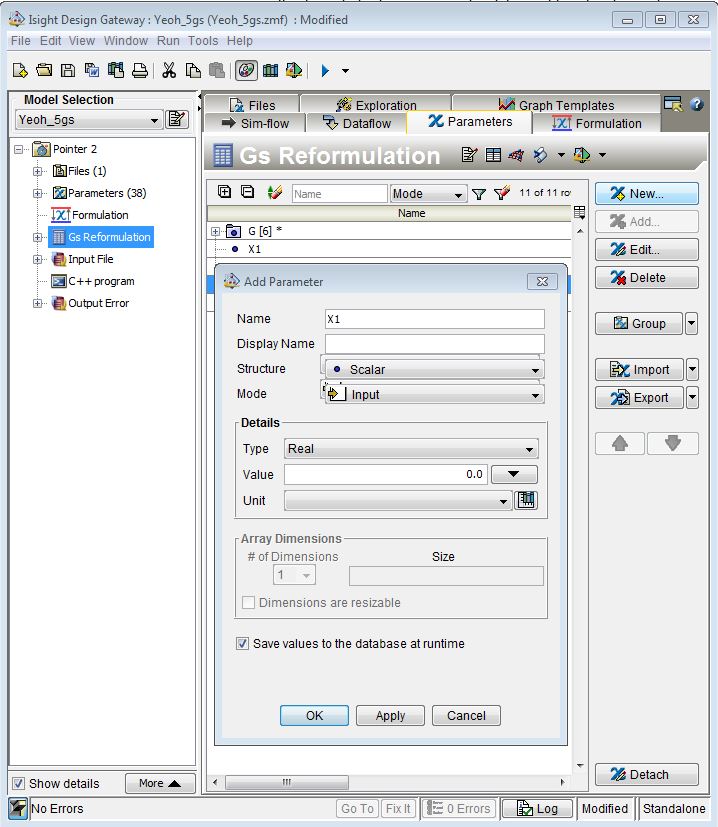


Figure 14- Create new variables

Inside the calculator configure the equations as follows:

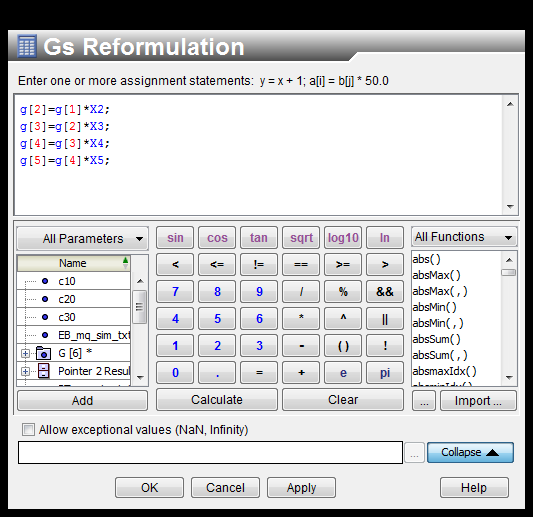


Figure 15- Implement the reformulation equations

Press always “Apply” to save the changes.

The last step to implement the reformulation is define X2, X3, X4, X5 as design variables inside the Exploration component and the lower and upper bounds set between.

C++ program OS Component

The OS command component is responsible to integrate the C++ program on the workflow. Basically, it contains the path to the executable, allowing Isight to execute the program. As a suggestion, define the path as follows:

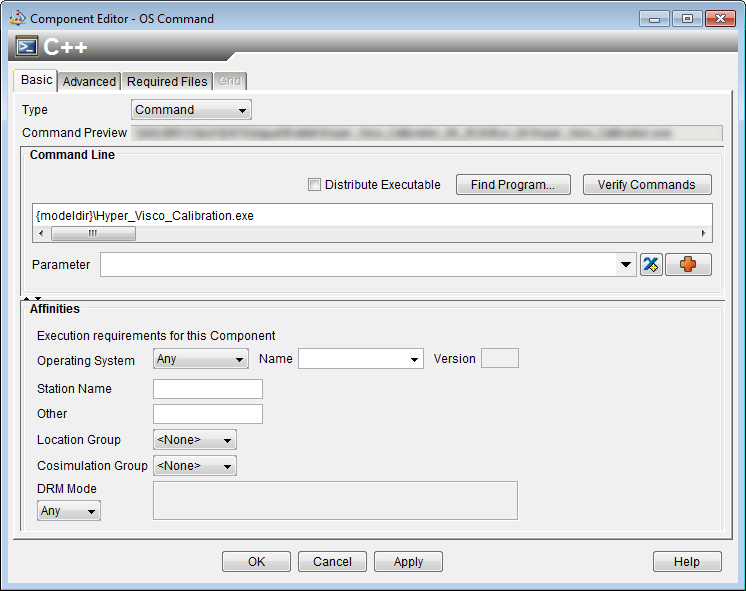
****

Figure 16- Configure OS command component

Output Error Data Exchanger

As already stated, the C++ program writes on each run the error measures for all data sets to the “Data\_suite\_error.txt” file. Isight, on the other hand, reads these values and uses them as optimization objectives. For this purpose a Data Exchanger needs to be set as follows:

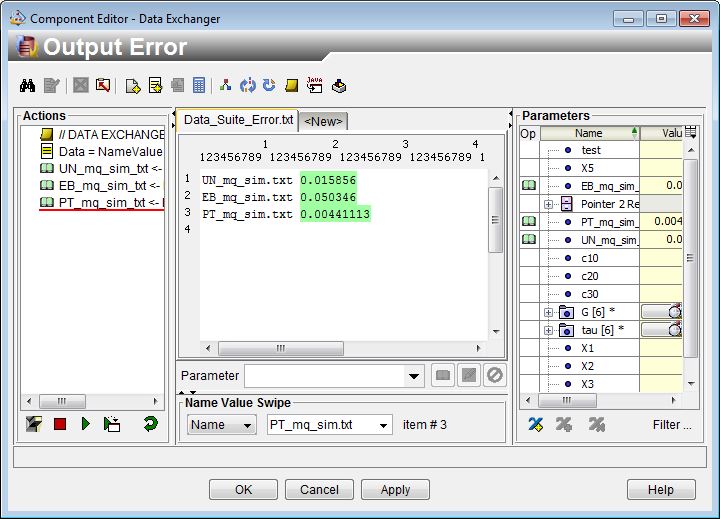


Figure 17- Configure the Output Error Data Exchanger

The user must define a variable for each file and set them as read variables. The fastest way to achieve this is, when selecting/configuring the file, to set the format to “Name/Value”. After configure this data exchanger set the file properties as showed on figure 12.

Exploration Component (Pointer 2)

The Exploration component is defined to use the Pointer 2 optimization method. This method revealed to be a good solution obtaining good results with different Yeoh material models configurations: 5, 4 and 3 Prony terms.

The most important options that need to be set on this component are the number of sub-flow runs, the design variables and its lower/upper bounds, and the optimization objectives.

On the next example the Pointer 2 is configured to 2000 sub-flow runs, 11 variables (4 Prony terms Yeoh model) with the objective of minimize each error value. Leave the other options as default.

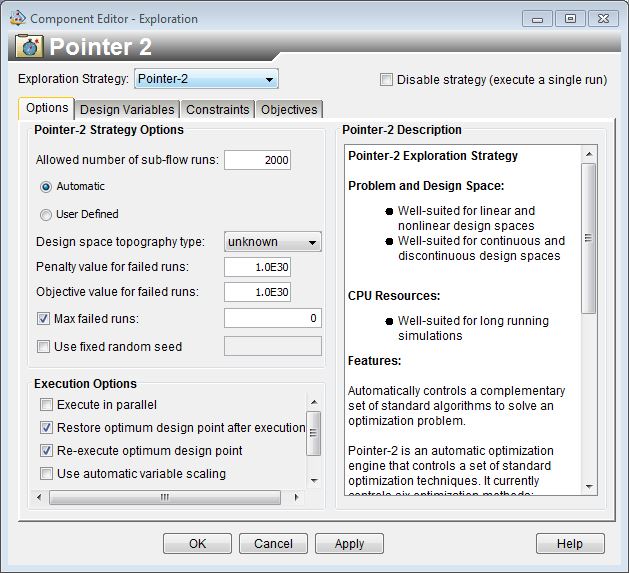


Figure 18- Defining the number of sub-flow runs

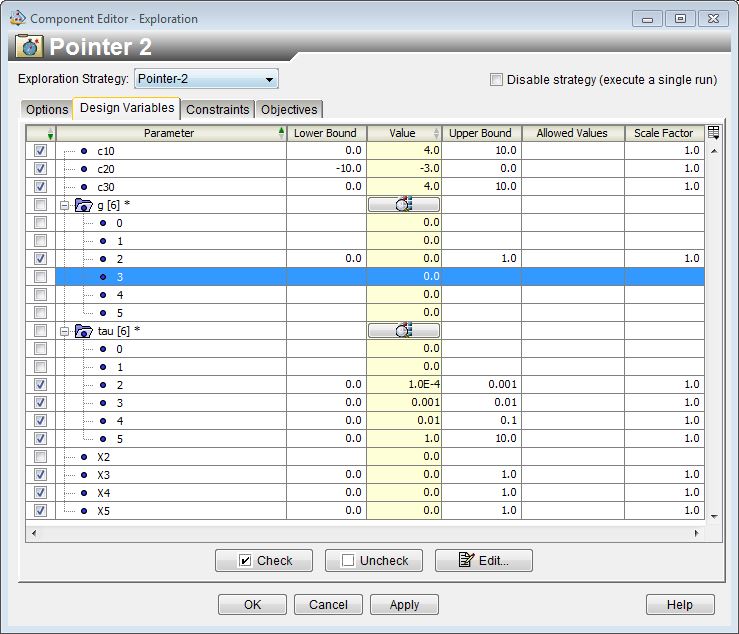


Figure 19- Yeoh Model with 4 Prony terms: Design Variables, ranges and starting points

**Important:** Note that the previous configuration of the calculator, figure 15, is for a 5 Prony terms model. For a 4 Prony terms model, g[2] needs to be set as design variable and the calculator component needs to be reconfigured by ignoring the g[2] dependency on g[1]. Comment the equation as shown on figure 20.

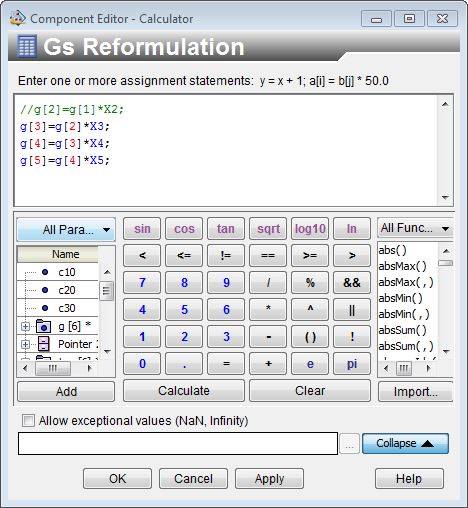


Figure 20- Calculator reconfiguration for a 4 Prony terms model

Otherwise the g[1] value will affect the design variable g[2] and consequently all the other Prony terms. Because g[1] will be deactivated and defined as 0.0, this causes a problem on the optimization.

Defining the design variables in terms of number, starting point and ranges is the most challenging part in terms of finding a good answer. During the experiments, The Pointer 2 showed to be less sensitive to the starting point, however a good initial guess is always important to help getting a good solution faster.

After multiple runs using different workflows some conclusions can be taken in terms of the behavior of the Yeoh Model hyper-viscoelastic calibration:

The C parameters have generally the same behavior. C10 and C30 are always positive values, while C20 is negative.

The bounds were arbitrary defined to a value of 10.0 for C10 and C30 upper bound, and -10.0 for C20 lower bound. These values were chosen because they are bigger/smaller enough to guarantee that the C parameters don’t reach the limits during simulation.

In terms of the influence on the curve fitting, the C10 is responsible for the stiffness of the simulation curves while C20 and C30 affect its curvatures. A good initial guess for C10 is the most important.

With 6 terms available, the user must active g[2], X3,X4,X5 and set the range between 0 and 1. A safe guess for the first g can be a value between 0 and 0.3.

In terms of taus, the user must active from tau[2] to tau[5], respecting the active Gs. Usually one tau per decade of time is defined with the lower bound set to 0 and the upper bound set to 10 times more than the starting point.

When running the workflow is possible to see the response of the model and understand from the results if the ranges are well defined or need to be readjusted, e.g., when a variable reaches the upper or lower bound before the simulation ends.

When available, the stress relaxation curve of the material can be a huge help to analyze the areas with more viscoelastic effect and “guessing” a good starting point.

To complete the Pointer 2 configuration the user must define as objectives minimize all the error values variables.

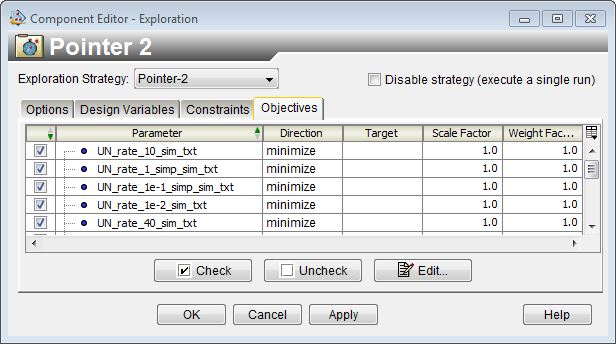


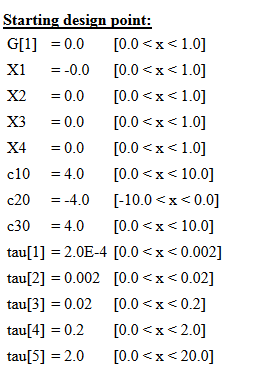
Figure 21- Pointer 2 optimization objectives

Note: For the user convenience, a folder called “Run\_Dir” includes everything set up, including the Isight model. The user can run directly from the folder or copy its content to a new one in his hard drive. The Isight model was set-up as shown above and should run without any problems.

Isight Results

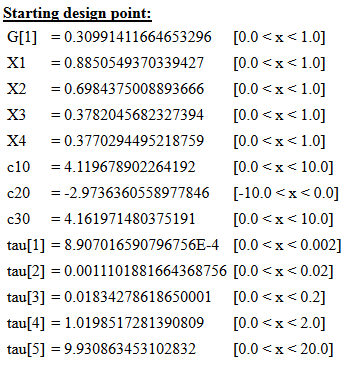
In this section some results are shown using the workflow described above with different Yeoh model configurations, resulting from varying the number of Prony terms. As characteristic from a material calibration problem these aren’t the right and unique answers.

Configuration 1: Yeoh model with 5 prony terms

On the first configuration a Yeoh model with 5 prony terms was defined with the starting point:

This optimization was concluded in two consecutive runs. After the results obtained on the first run, it was decided to restart the optimization from the last optimum design point. This decision was based on the fact that the best solution was obtained at iteration 1992/2000. Being almost at the end of run, this factor can be an indicator that the solution can be improved.

For the second run the starting point was:



On the second run, the optimum design point was obtained at iteration 1792/2000. In total, the all optimization took approximately 12 minutes. After concluded, the optimum design point and error measures obtained were:

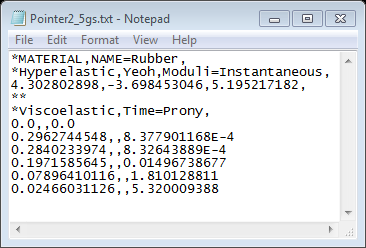


Figure 22- Optimum design point

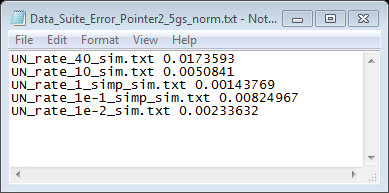


Figure 23- Optimum design point error measures

Looking to the final tau values, tau[1] and tau[2] are very close, while tau[4] and tau[5] are in the same decade of time. This solution suggests a curve fitting with a simpler model reducing the prony terms to 4 or even 3.

This model can be directly simplified to a 4 Prony terms solution summing g[1] and g[2] and corresponding the result to a new tau that results from the mean of tau[1] and tau[2]:

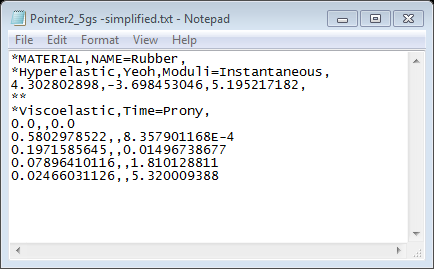


Figure 24- Equivalent Optimum design solution

Graphically the curve fitting obtained with configuration 1 looks like:

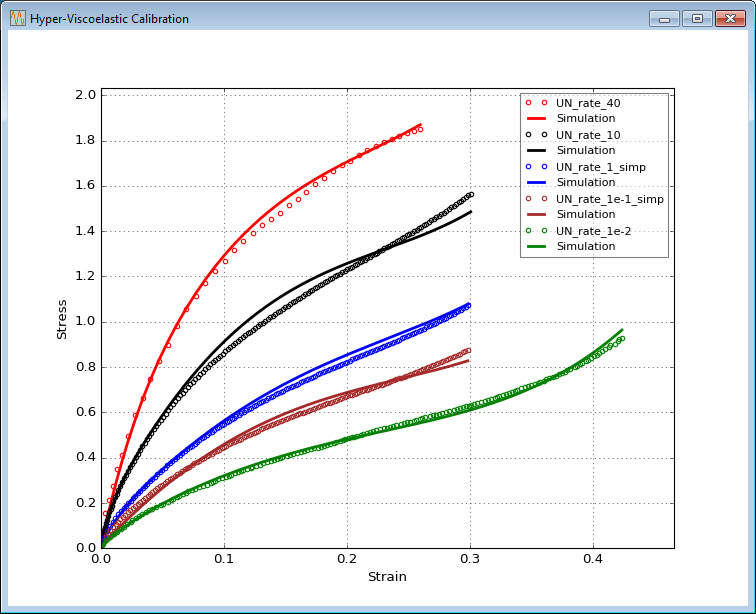
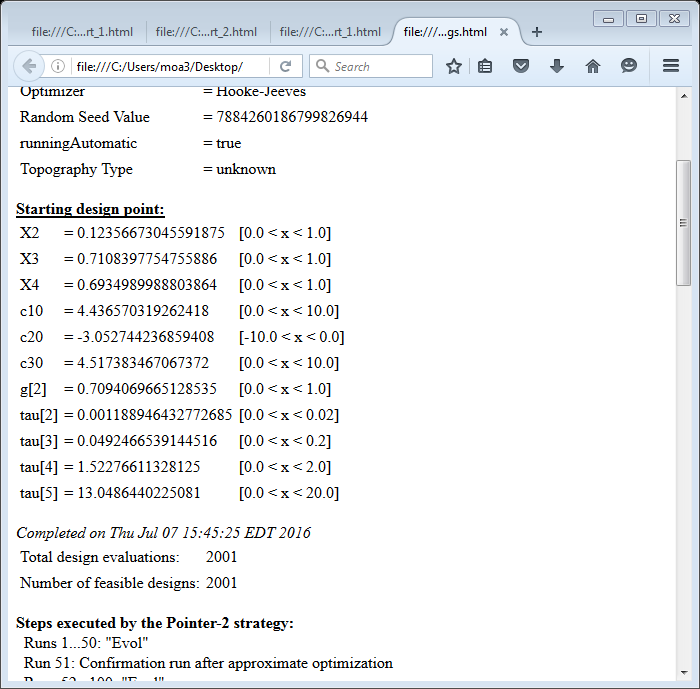


Figure 25- Configuration 1 curve fitting final results

Configuration 2: Yeoh model with 4 Prony terms

In this case a Yeoh model with 4 prony terms is configured. The starting point is similar to the restarting point of the previous configuration without the lowest tau.



This run took around 6 minutes to conclude and resulted in the optimum design point

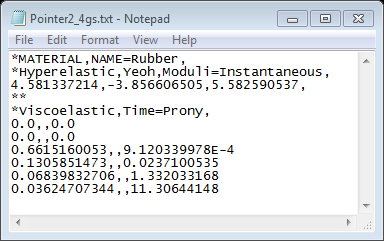


Figure 26-Configuration 2 Optimum design point

Comparing the final results with the previous model the parameters values are similar, showing some consistency in terms of final solution. The major difference is in the last tau that is almost twice the value of the first solution. In terms of error measures, the biggest changes occurred in the curve fitting of the “UN\_rate\_10\_sim.txt” where the error measure decreased more substantially.

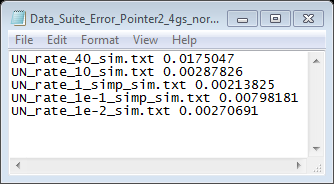


Figure 27- Configuration 2 error measures

Graphically the curve fitting obtained with configuration 2 looks like:

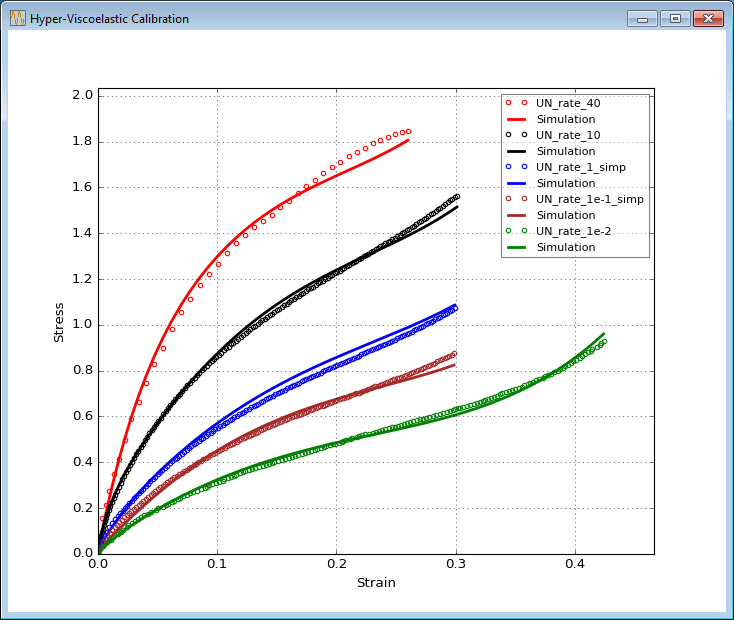
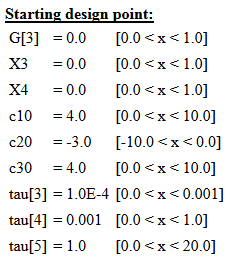


Figure 28-Configuration 2 curve fitting final results

Configuration 3: Yeoh model with 3 Prony terms

On previous configurations the results show that the biggest tau values are on the same decade of time. This fact makes reasonable to consider curve fitting with a Yeoh model with 3 Prony terms. In terms of the starting point, the tau values were readjusted based on the previous results for the smallest and biggest tau, while the range of the middle tau was extended to cover 2 different decades.



Looking to the obtained results, the values show again some consistency compared to previous results. The Yeoh model C parameters have similar values while, in terms of the Prony series the most significant changes were on g[3]. It is interesting that its final value is approximatly equal to the sum of the two smallest gs of the previous configurations.

The wide range defined for the second tau resulted in its value increased by one decade ending up being in the same decade as the previous configurations.

In terms of fitting the experimental data, the results are a little worst duo to the less number of prony terms used. However, the differences are not so relevant and this configuration still produces a good fit with a minimun amount of prony terms.

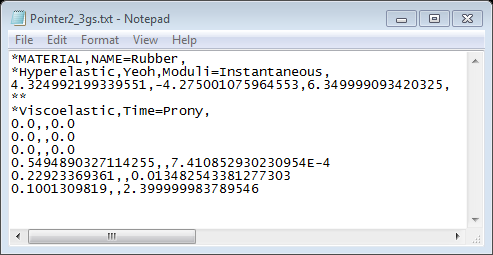


Figure 29- Configuration 3 Optimum design point

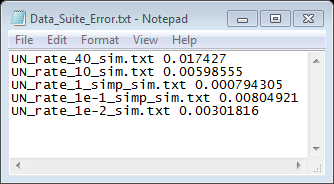


Figure 30- Configuration 3 Error measures

Graphically the results obtained with configuration 3 looks like:

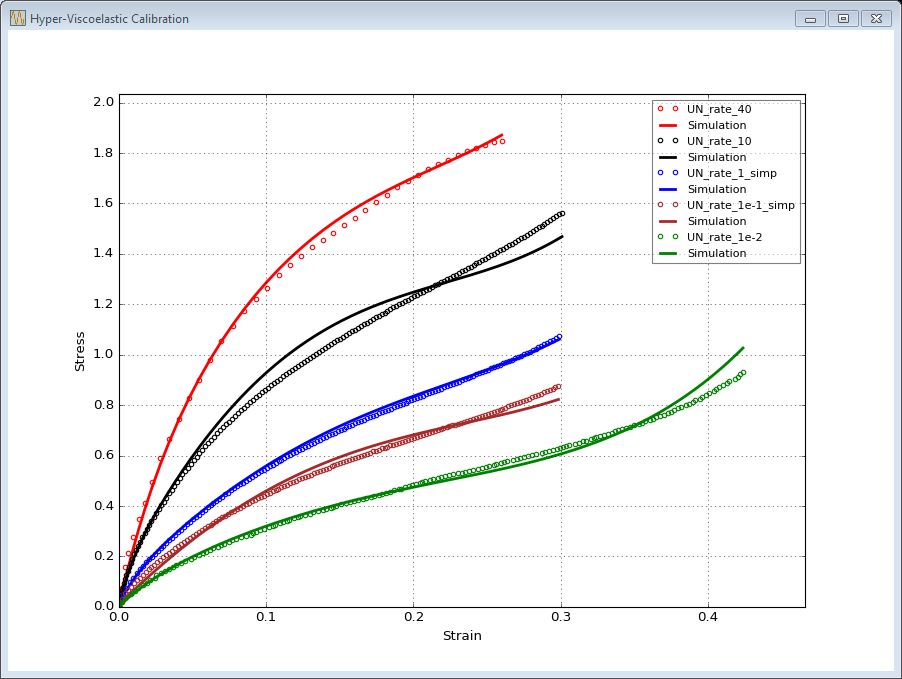


Figure 31- Configuration 3 curve fitting results

Isight Model Behavior

Pointer 2 is a powerful optimizer that combines multiple algorithms to obtain a final solution. It is designed to automatically adapt and decide which optimization sequence is the best for the specific problem in a kind of self-learning capability. From the tests performed with the current model, the convergence to a good solution seems to be faster and more consistent compared to the use of a single optimization process.

However, because of the way that Pointer 2 works, it’s difficult to replicate a result using the same starting point. Some runs were made using the same starting point and the option “Use fixed random seed” to replicate an existing solution. Results have shown that this model didn’t converge always to a similar solution. The algorithm was able to find solutions with similar objective values but different parameters. In terms of the convergence time, solutions were obtained with a maximum of two runs. However, in some attempts the Pointer 2 method was able to converge twice as faster, giving a good answer after one run.

To perform a second run starting from the best result of the first, the user has to right click over the green highlighted solution and select “Init model from this run”.

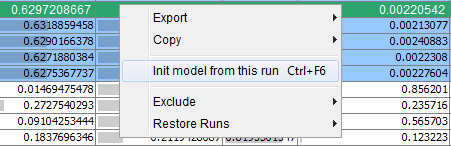


Figure 32- Restart model from a selected point

Appendix: Code Blocks Settings

The C++ program was built using the IDE Code Blocks version 16.01. Before starting using the IDE the compiler needs to be configured under “*Settings*- *Compiler*…”

First define the type of compiler to GNU GCC Compiler and active the **“-std=c++14**” option:

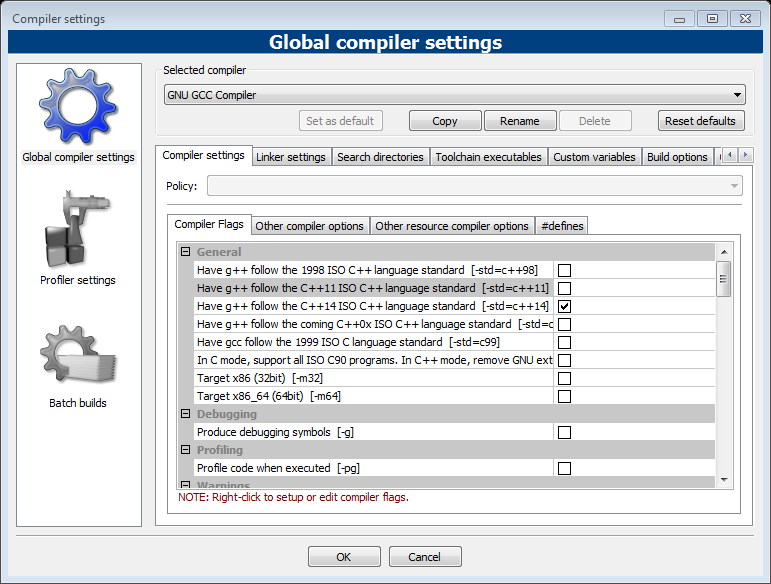


Figure 33- Compiler Flags

Under “Compiler settings – Other compiler options” define the options as below:

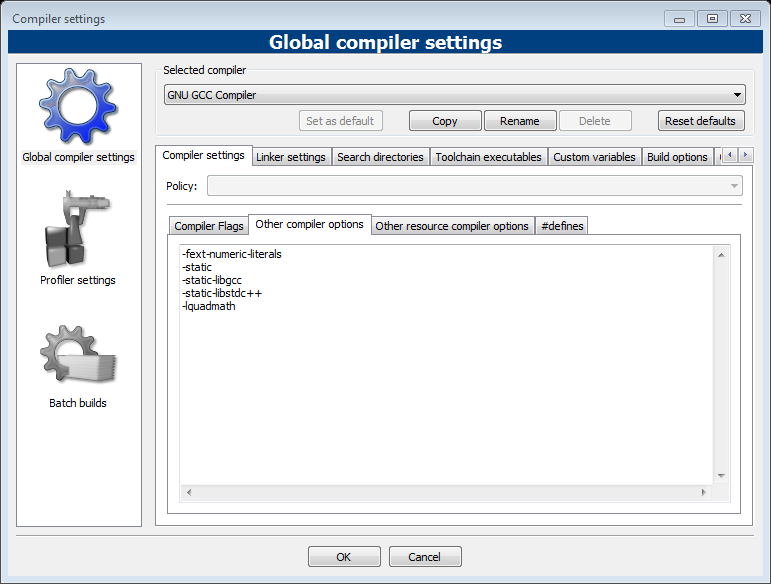


Figure 34- Compiler Options

On the “Linker settings” define the linker options as below:

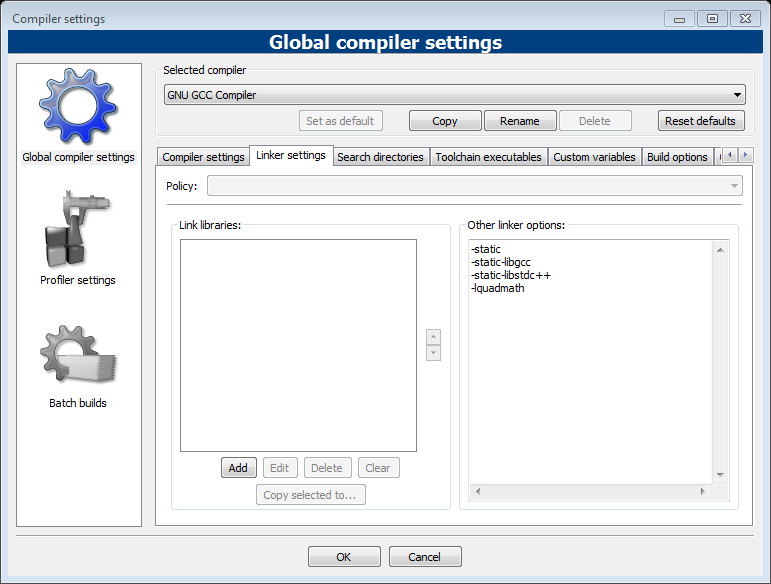


Figure 35- Linker settings

Define the path to the compiler’s directory and set all the program files using the “…” button:

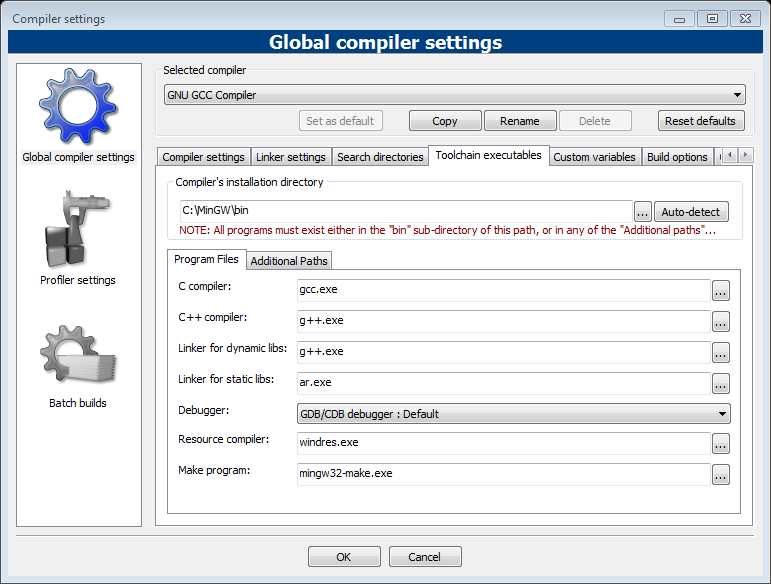


Figure 36- Toolchain executables

After setting the compiler, create a working directory for the project with all the source code files. In Code Blocks, go to “File- New- Project”, select ‘Console application” and press “Go”.

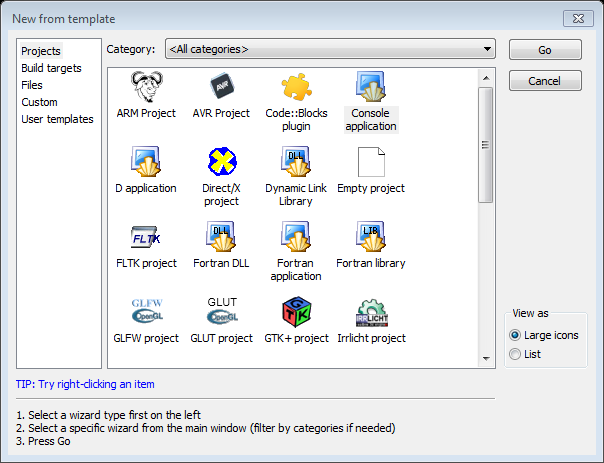


Figure 37- Define project type

Select the language to C++ and press “Next”.

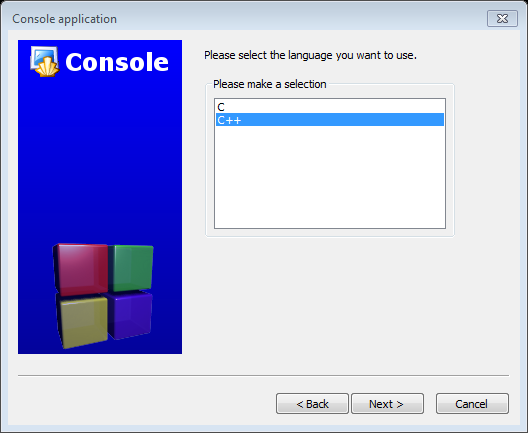


Figure 38- Define the language

Configure the project title and the folder to create the project in:

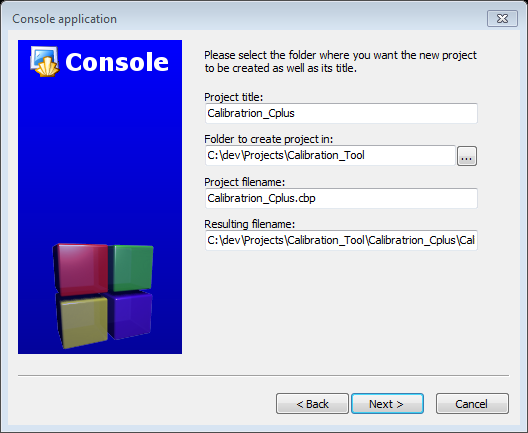


Figure 39- Define the project name and folder

Define the compiler settings as follow, and press “Finish” to create the project:

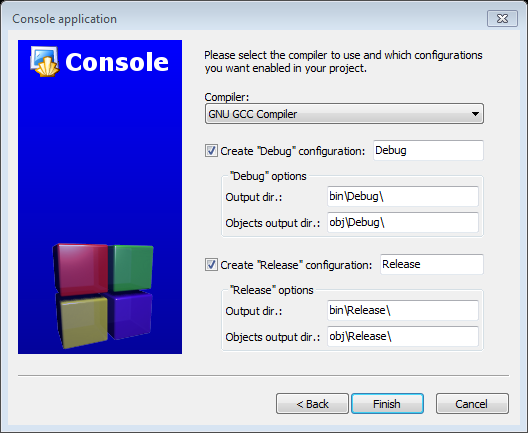


Figure 40- Compiler settings

A new project should be created and will look like this:

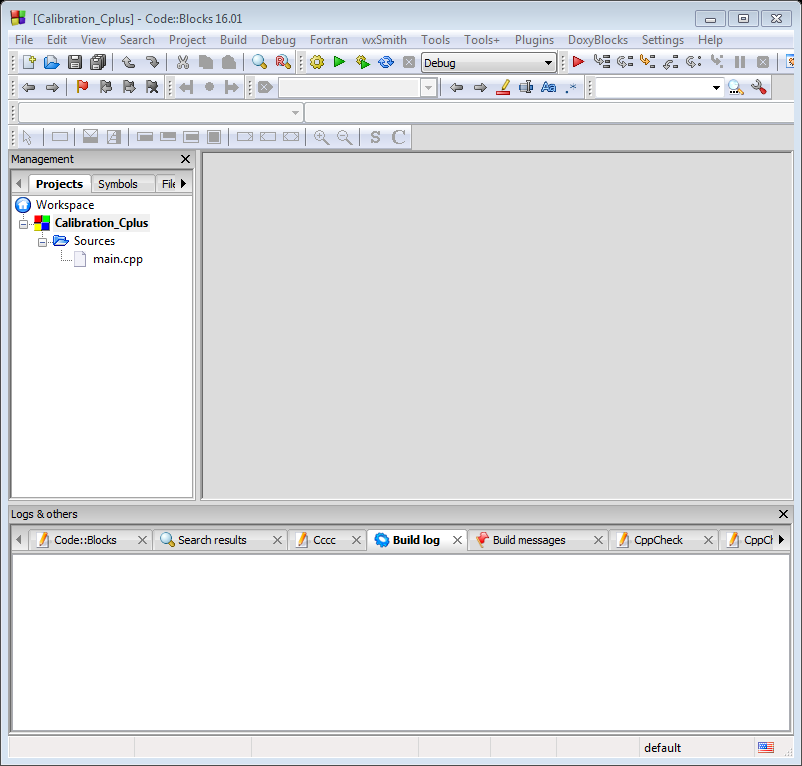


Figure 41- New project template

Go to the project folder and paste all the source code files. When asked, choose to “Copy and Replace” the existing main.cpp file.

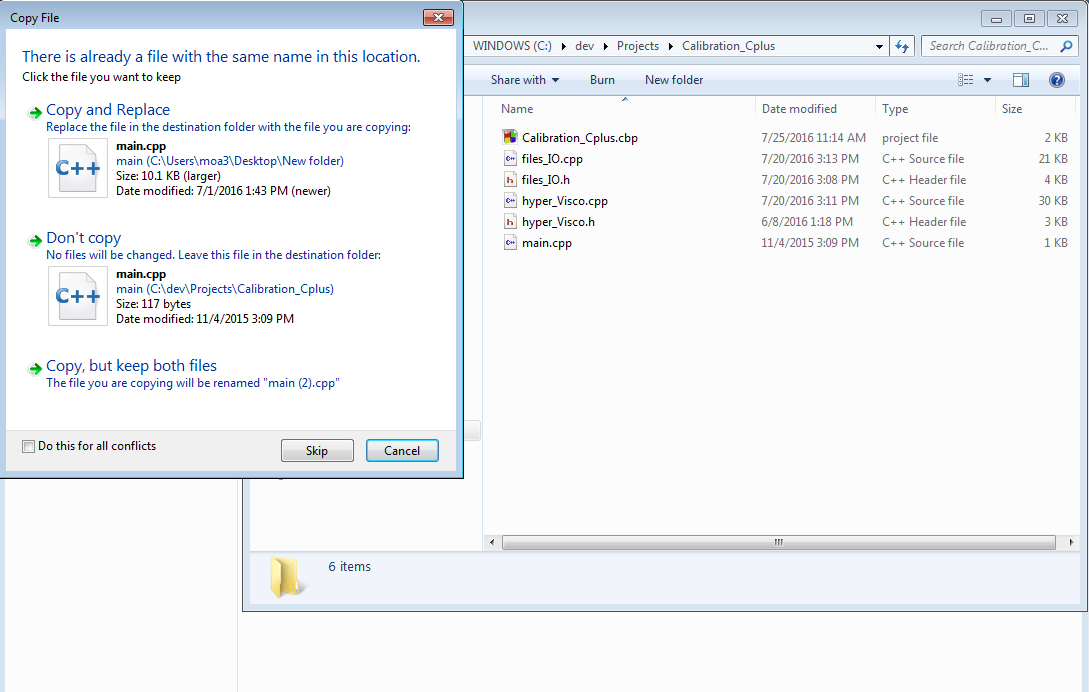


Figure 42- Copy & paste source code

Return to Code Blocks and go to “File- Open…”. Select all the source code files inside the directory and press “Open”. Add all the files to the project by selecting each file and choose the right mouse click option “Add file to the active project”. On the next pop-up accept the default settings by pressing “Ok”.

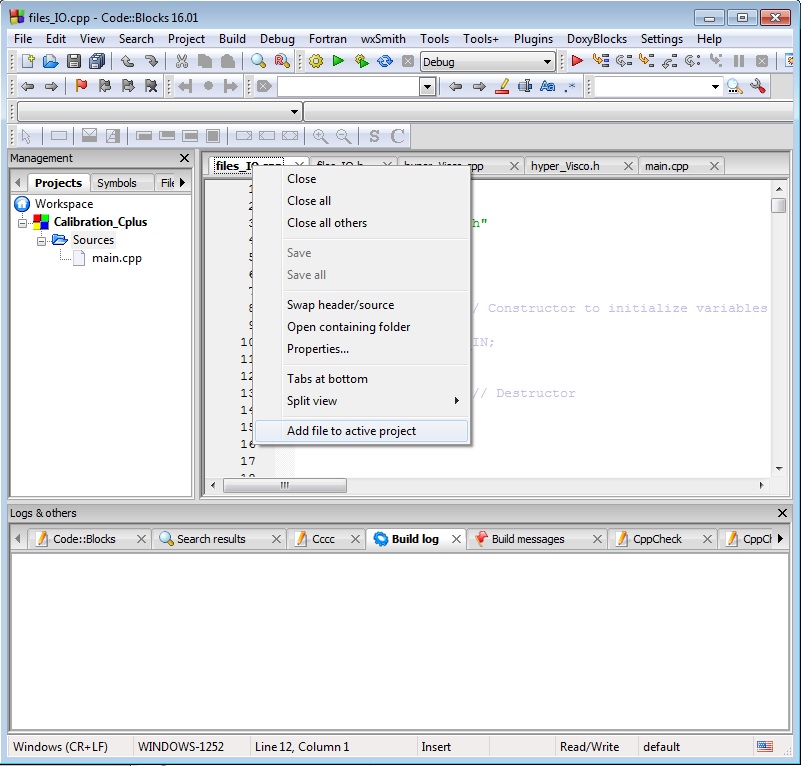


Figure 43- Add files to the project

After completing this step the project should look similar to the image below:

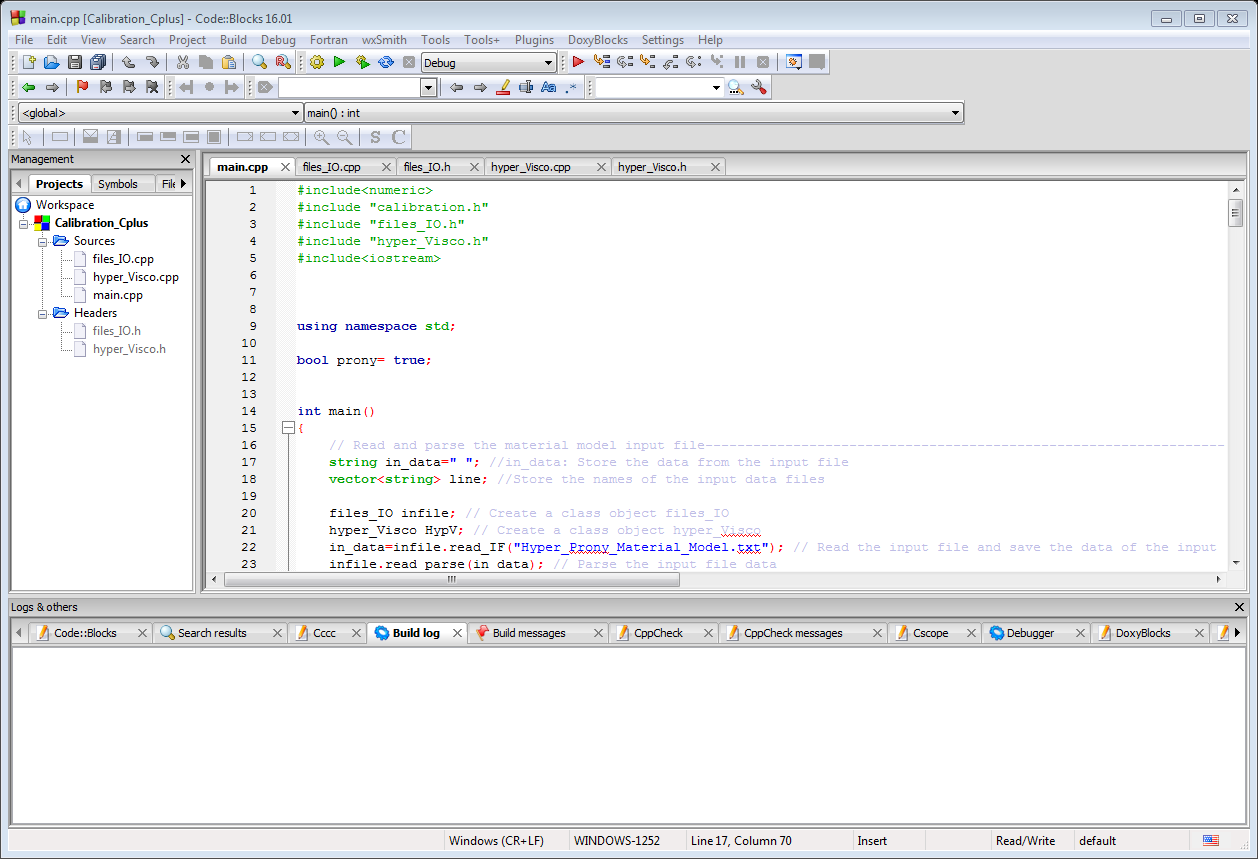
****

Figure 44- Project with all the files

Press the Build and Run button **** or F9 key to compile the project. It should compile successfully and open a command prompt after the action is concluded. However, the program will not execute correctly duo to the missing files “Hyper\_Prony\_Material\_Model.txt”, “Data\_Suite.txt” and the test data files. Copy these files to the project folder and compile the project again. This time it should compile and run correctly.