DEFACTO - battery DEsign and manuFACTuring Optimisation through multiphysic modelling

D.6.2

Date: 30.06.2022

This document is a description of the DEFACTO D6.2 deliverable (contract no. 875247 coordinated by CIDETEC). This document explains the details concerning the release of the DEFACTO time- and parameter-adaptive ROM optimisation tool (based on the DEFACTO p4D reduced order model, also developed in the framework of the DEFACTO project) under a Free Open-Source Software (FOSS) license, which constitutes the deliverable D6.2 itself. In the document, the present capabilities of this tool and their extension in the planned updates are described. Some information concerning the used FOSS license and the distribution of the software using the project web page is included as well. Finally, a glimpse on the use of this software tool and some comments on the documentation provided are presented in the document.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247.

Project details

Project acronym	DEFACTO	**Start / Duration**	*01/01/2020 (42 months)*
Topic	LC-BAT-6-2019	**Call identifier**	*H2020-LC-BAT-2019-2020*
Type of Action	RIA	**Coordinator**	*CIDETEC*
Contact persons	*Elixabete Ayerbe*		
Website	**www.defacto-project.eu**		

Deliverable details

Number	*D6.2*		
Title	*Release of the time and parameter-adaptive ROM optimisation tool under FOSS*		
Work Package	*WP6 – Optimisation and sensitivity analysis*		
Dissemination level	Public	**Nature**	*Other (Software)*
Due date (M)	M30	**Submission date (M)**	*M30*
Deliverable responsible	UPM	**Contact person**	*Fernando Varas*

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

Deliverable Contributors

	Name	Organisation	Role / Title	*E-mail*
Deliverable leader	Fernando Varas	UPM	Partner	*fernando.varas@upm.es*
Contributing Author(s)	María Higuera	UPM	Partner	*maria.higuera@upm.es*
	Eduardo Jané	UPM	Partner	*eduardo.jane.soler@upm.es*
	Ruth Medeiros	UPM	Partner	*ruth.medeiros@upm.es*
	Rubén Parra	CID	Partner	*rparra@cidetec.es*
	Fernando Varas	UPM	Partner	*fernando.varas@upm.es*
Reviewer(s)	Morgan Cameron	ESI	WP6 leader	*morgan.cameron@esi-group.com*
Final review and quality approval	María Yáñez & Elixabete Ayerbe	CID	Project Coordinator	*myanez@cidetec.es* *eayerbe@cidetec.es*

Document History

Date	Version	Name	Changes
11.07.2022	1.0	UPM	Initial draft
12.07.2022	1.1	CID	Brief revision on format by PC
15.07.2022	2.0	ESI	Revision by WP leader
22.07.2022	2.1	UPM	Implementation of revision comments
22.07.2022	3.0	CID	Final revision

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

Content

1 LIST OF FIGURES 5

2 ACRONYMS AND ABBREVIATIONS 5

3 EXECUTIVE SUMMARY 6

4 INTRODUCTION 7

5 FEATURES OF THE RELEASED SOFTWARE 7

5.1 Features of cideMOD component 7

5.2 Features of echROM component 8

5.3 Features of OptiBat component 9

6 FREE OPEN-SOURCE SOFTWARE LICENSE 9

7 DISTRIBUTION THROUGH THE PROJECT WEBSITE 11

7.1 Software section in DEFACTO project website 11

7.2 Software packaging tool 11

8 SOFTWARE INSTALLATION 13

9 AN APPLICATION EXAMPLE CASE 13

10 SOFTWARE DOCUMENTATION 17

11 CONCLUSIONS 19

12 BIBLIOGRAPHY 20

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

1 List of Figures

Figure 1. Screenshot of a preliminary design of the new section in DEFACTO website 12
Figure 2. opt_dict.json file example 15
Figure 3. Battery cell mesh 15
Figure 4. Python code screenshot showing the optimisation problem resolution 16
Figure 5. Objective function value along the optimisation process. 16
Figure 6. Example of a commented code (top image) and its corresponding documentation interface automatically generated by Sphinx tool (bottom image). 18

2 Acronyms and abbreviations

AGLP	Affero General Public License
DFN	Doyle-Fuller-Newman (model)
ERC	European Research Council
FOM	Full Order Model
FOSS	Free Open-Source Software
FSF	Free Software Foundation
GPL	General Public License
Gr	Graphite
LGLP	Lesser General Public License
NMC	Nickel Manganese Cobalt Oxide
p2D	Pseudo two-dimensional
p3D	Pseudo three-dimensional
p4D	Pseudo four-dimensional
ROM	Reduced Order Model
SEI	Solid Electrolyte Interface

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

3 Executive Summary

In the framework of DEFACTO work package 6, an optimisation tool based on a time and parameter-adaptive reduced order model has been developed. As stated in DEFACTO grant agreement, the software implementing this p4D optimisation tool is being released under a Free Open-Source Software (FOSS) license. Deliverable D6.2 corresponds precisely to the first release of this software.

This software is based on three tools. The first tool (called cideMOD and developed in WP5) implements a continuum pseudo four-dimensional model (p4D) for a battery cell. The second one (called echROM and developed in WP6) corresponds to an acceleration software tool based on (adaptive) model order reduction techniques and designed to (significantly) reduce the computational effort needed to solve pseudo four-dimensional battery models. The last tool (called OptiBat) allows to efficiently optimise the design of a battery cell using time-adaptive and parameter-adaptive model order reduction techniques, making use of the updated versions of cideMOD and echROM software tools (a first version of these two tools was released in deliverable D6.1).

With this initial release, an optimisation template is also distributed. Using this template, the energy delivered by a given battery cell at a constant C-rate discharge can be maximised by optimising the value of anode and cathode porosity and thickness. In later software releases, additional templates will be distributed, where more optimisation parameters and constraints will be considered.

The software is released using the GNU Affero General Public License version 3. This Free Open-Source Software (FOSS) license makes sure that any derivative work (including those related to cloud computing exploitation of the code) preserves the same freedoms offered now to any user of the released code.

The initial release of the code is made through the DEFACTO project website. To do this, a new section is being added to the website. To make software installation easier, all tools (cideMOD, echROM and OptiBat) are distributed together, and they are also bundled with other FOSS tools used by them: FEniCS finite element library, multiphenics extension and Gmsh meshing tool (as well as a Python interpreter and NumPy and SciPy packages). A single file corresponding to a Docker image is being hosted at the website. Detailed installation instructions of this image are available at the same web page. Thanks to the Docker platform technology, after image deployment all software tools are completely installed and configured.

Software tools are distributed with three classes of documents: a theory manual, a developer manual and a collection of tutorials. Those documents are also bundled in the Docker image and available after software installation. Each documentation section will be enriched along DEFACTO project execution although documents in the present, initial release are already fully functional.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

4 Introduction

Deliverable D6.2 corresponds to the release of a software tool able to efficiently optimise the design of a battery cell using a time-adaptive and parameter-adaptive reduced order model. This document contains a description of the software tool which is available at the DEFACTO project website. According to the DEFACTO proposal this software tool is released under a Free Open-Source Software (FOSS) license.

The released software is internally organized in three components:

- The first one, called cideMOD, is being developed in the framework of WP5. This tool corresponds to a simulation software able to solve p2D, p3D and p4D continuum models. It is based on the FEniCS finite element library [1] and an extension of this library called multiphenics [2] (developed with the support of the European Research Council, ERC). The meshing tool Gmsh [3] is also used. FEniCS library, multiphenics extension and Gmsh meshing tool are distributed under FOSS licenses.
- The second component, called echROM, provides an acceleration tool for the first one developed in DEFACTO WP6. Based on time-adaptive model order reduction techniques [4] [5], the solution of a p4D continuum model is carried out combining the use of cideMOD for some short periods of time and the integration of the p4D model by a reduced order model (ROM) built on-the-fly (using information from the solution provided by cideMOD).
- The third component, called OptiBat, is a Python package developed in DEFACTO WP6 to efficiently perform a cell optimisation. In order to hugely reduce the computational cost required during the cell optimisation process, this tool combines the use of the time-adaptive DEFACTO p4D ROM (released as Deliverable D6.1 and composed by the two previous tools) and a new parameter-adaptive strategy.

These software components are being continuously developed throughout the DEFACTO project, extending its present capabilities (explained in the next section). Nevertheless, the present release already contains a fully functional simulation environment for electrochemical models. Future software updates will incorporate new models and simulation acceleration improvements.

5 Features of the released software

5.1 Features of cideMOD component

The cideMOD library, released within Deliverable D6.1, is a generalization of the Doyle-Fuller-Newman [6] approach for the modelling of lithium-ion battery cells to work with 1D, 2D or 3D cell geometries. In the DEFACTO project the focus is put on the 3D geometry and therefore in the p4D model. The latter implements porous electrode theory and dilute concentration theory to simulate the electrochemical performance of battery cells considering each domain as a continuum. Through this approach, the performance of a real geometry of the cell can be simulated in a reasonable amount of time.

The library is built on top of FEniCS, using it as the underlying finite element and automatic differentiation engine. The cideMOD library allows the simulation of different battery configurations with arbitrary materials in a single interface. Making use of Gmsh meshing software, different battery shapes can be created and simulated including different tab configurations. It also comes with a

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

simplified interface to run automatically arbitrary testing conditions, cycling protocols and usage profiles. Some of the features of the p4D model are summarized here:

- General features:
  - Customizable pouch cell geometry and tab position.
  - Highly customizable simulation conditions.
  - Portable input/output format (json/[xdmf-txt]).
- Electrochemical model:
  - Support for electrodes with several active materials
  - Nonlinear transport properties
- Thermal model:
  - Takes into account major heat sources.
  - Temperature dependent transport properties (Arrhenius relation).
- Electrochemical ageing models:
  - SEI growth: Diffusion limited SEI growth model inspired in Safari et al. (2009) [7].

5.2 Features of echROM component

As previously described, the goal of the echROM component, also released within Deliverable D6.1, is the acceleration of (computationally demanding) p4D simulations with cideMOD software using adaptive model order reduction techniques.

According to this goal, the evolution of the echROM component closely follows cideMOD component development along DEFACTO project execution. At this stage, only the electrochemical and thermal models can be accelerated. Ageing/damage models, already available in p4D, will be included in echROM acceleration tool very soon.

The software design is intended to provide an acceleration tool as transparent (to the final user) as possible, as decided during software specifications preparation at the beginning of the project. Therefore, only a minimal interaction concerning algorithm parameter selection is required if the user decides to accept default parameter selection. The echROM component can be used providing exclusively the JSON input file needed to simulate the battery and the JSON input file containing the adaptation technique parameters. In this case, all parameters related to model order reduction techniques are automatically selected. Nevertheless, more advanced users can gain a complete control of the algorithm implementation instead. But even in this case, an effort has been made to make the algorithm modification as easy as possible. According to this guideline, all the main aspects of the (adaptive) model order reduction algorithm are implemented in an easy-to-read Python script (only implementation details are hidden in functions).

Concerning output file formats, since conversion of the reduced order model solution to full order model format is quite easy, all available formats in cideMOD are being implemented.

Algorithmic acceleration already provides an outstanding speed-up for the reduced order model time integration using a Python implementation, since a rather low dimensional model is to be integrated (in contrast with very large models integrated by p4D). Nevertheless, an additional speed-up (up to two orders of magnitude) in the time integration of the reduced order model is achieved by the use of a Fortran library containing all the basic operations carried out during time integration of the reduced order model. As a consequence, the computational effort dedicated to the time integration of the reduced order model is almost negligible and the speed-up is approximately given by the fraction of the simulation time corresponding to the full order model by cideMOD.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

In order to obtain additional acceleration, information from a simulation database (if available) can be introduced in the adaptive model order reduction technique. This possibility will be used in the framework of the optimisation iterative algorithms, as shown below.

5.3 Features of OptiBat component

As previously described, the goal of the OptiBat component is the acceleration of the battery cell design optimisation process using the time-adaptive DEFACTO p4D ROM and a parameter-adaptation technique.

According to this objective, the evolution of OptiBat component closely follows the development of the cideMOD and echROM components throughout the DEFACTO project execution to adapt itself to their capabilities. At this stage, only the electrochemical model is considered to perform the battery cell optimisation. Thus, the software provides all the needed ingredients to carry out the optimisation of any objective function, considering exclusively electrochemical parameters. In particular, the tool supplies a template to maximise the delivered energy of a given battery cell at a constant C-rate discharge. Additional templates will be provided in the next release.

Moreover, the template distributed with this release can accommodate the optimisation of any combination of the following parameters: anode and cathode porosity and thickness. In addition, the OptiBat tool has the flexibility to impose (or not) some constraints to the problem. The current available optimisation constraints are:

- Inequality constraints: upper and lower bounds for the different considered parameters.
- Equality constraints
    - Imposition of a cell balance.
    - Imposition of the cathode capacity value.

In the same way as for the echROM tool, the design of the optimisation tool is intended to be as transparent as possible to the final user. Therefore, it is possible to use the tool with a minimal interaction concerning the optimisation algorithm. Thus, the OptiBat component can be used providing exclusively the input files needed by the echROM component and an additional JSON file containing the needed parameters to perform the optimisation (e.g. optimisation parameters and constraints).

6 Free Open-Source Software license

The DEFACTO project established the release of the developed software using a Free Open-Source Software (FOSS) license. The rationale for this license choice was the declared intention of the DEFACTO project to provide a useful tool for battery and electrochemical cell design available to any European manufacturer or research laboratory. At the same time, adoption of this software by a (hopefully large) community of researchers will ensure that this software can benefit from contributions by other users.

Concerning the selection of a particular FOSS license to release developed software tool, the following criteria were considered:

- A well-established license was preferred
- Since derived software (arising from software modification by community of users) is intended to be available to any user, a copyleft was selected

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

- Not only software distribution by third parties needed to be considered by the FOSS license, but also cloud computing software exploitation aspects were covered by the license

Additionally, the selected FOSS license needed to be compatible with the license used in FEniCS library (GNU LGPLv3 license), multiphenics extension (LGPLv3 as well), Gmsh (GPLv2), cideMOD (AGPLv3) and echROM (AGPLv3). Finally, the same FOSS license used for both cideMOD and echROM is used for the OptiBat tool, to simplify the software distribution.

According to these criteria, the GNU Affero General Public License version 3 (AGPLv3 in short, and endorsed by the Free Software Foundation) was selected. The short description of this license by the Free Software Foundation (FSF) follows:

> *This is a free software, copyleft license. Its terms effectively consist of the terms of GPLv3, with an additional paragraph in section 13 to allow users who interact with the licensed software over a network to receive the source for that program. We recommend that developers consider using the GNU AGPL for any software which will commonly be run over a network.*

In particular, license section 13 quoted in the previous description reads:

> *13. Remote Network Interaction; Use with the GNU General Public License.*
>
> *Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.*
>
> *Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part, which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.*

As stated in section 3, GNU Affero GPL license makes sure that not only software distribution/exploitation through communication of the (modified or unmodified) code itself is covered by FOSS provisions but also software exploitation through other services, such as cloud computing platforms, is covered by these principles. Thus, any (present or future) user of these pieces of software will benefit from any modification/improvement included in any form of software derivative.

On the other hand, it must be highlighted that in the present software release FEniCS, multiphenics and Gmsh are distributed without any modification, but FOSS license used to distribute these software items makes it possible to modify the code if this is needed in any future software update.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

7 Distribution through the project website

7.1 Software section in DEFACTO project website

Concerning software distribution, it was decided to use the DEFACTO project website to host initial software releases. This decision was made for several reasons. Firstly, it was intended to make clear to any software user both the origin of the software in the framework of DEFACTO project and the strong link of the developed models to the project work packages. At the same time, software diffusion will benefit from website visits and other DEFACTO dissemination activities (generating Internet traffic to the project website), and software updates schedule and contents can be anticipated from DEFACTO project plan. Before DEFACTO project finalization, a decision will be made on the software hosting after the end of the project.

To this end, the section titled 'Software' was used in the DEFACTO project to distribute the software, as was done for the time-adaptive DEFACTO p4D ROM. Figure 1 shows a screenshot of a preliminary design of the presentation of the **DEFACTO time- and parameter-adaptive ROM optimisation tool** on the project website.

7.2 Software packaging tool

It was decided to make software distribution of the developed tools as easy as possible, in order to promote a wider dissemination. Thus, instead of releasing exclusively the **DEFACTO time- and parameter-adaptive ROM optimisation tool** (forcing final users to also install FEniCS, multiphenics and Gmsh, as well as a Python interpreter, by themselves), it was decided to distribute all the needed software together. At the same time, avoiding possible version incompatibilities was also of paramount importance.

With the previous considerations in mind, it was decided to distribute all the software items using Docker technology. Docker makes very easy both to bundle software tools (into a single image) and to configure these tools upon installation. Since files created using Docker technology to distribute software (these files are usually called containers) already include a small operating system kernel, the same file is used to distribute the software to be run on different environments/operating systems. Docker container created using the Docker image available at DEFACTO website can thus be used to run the p4D optimisation tool on Linux, Windows and macOS computers. Software execution using Docker provides a less computational resource demanding alternative (from the final user perspective) than a virtual machine approach, which is relevant since p4D battery cell optimisation itself will require expensive computations. Experienced users requiring intensive execution of the p4D optimisation tool can choose to directly deploy both software tools in their workstations or computation servers (thus avoiding Docker overloading, which is nevertheless really small in GNU Linux machines). When commercial organisations do not allow the installation of Docker, the different software components can also be directly install.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

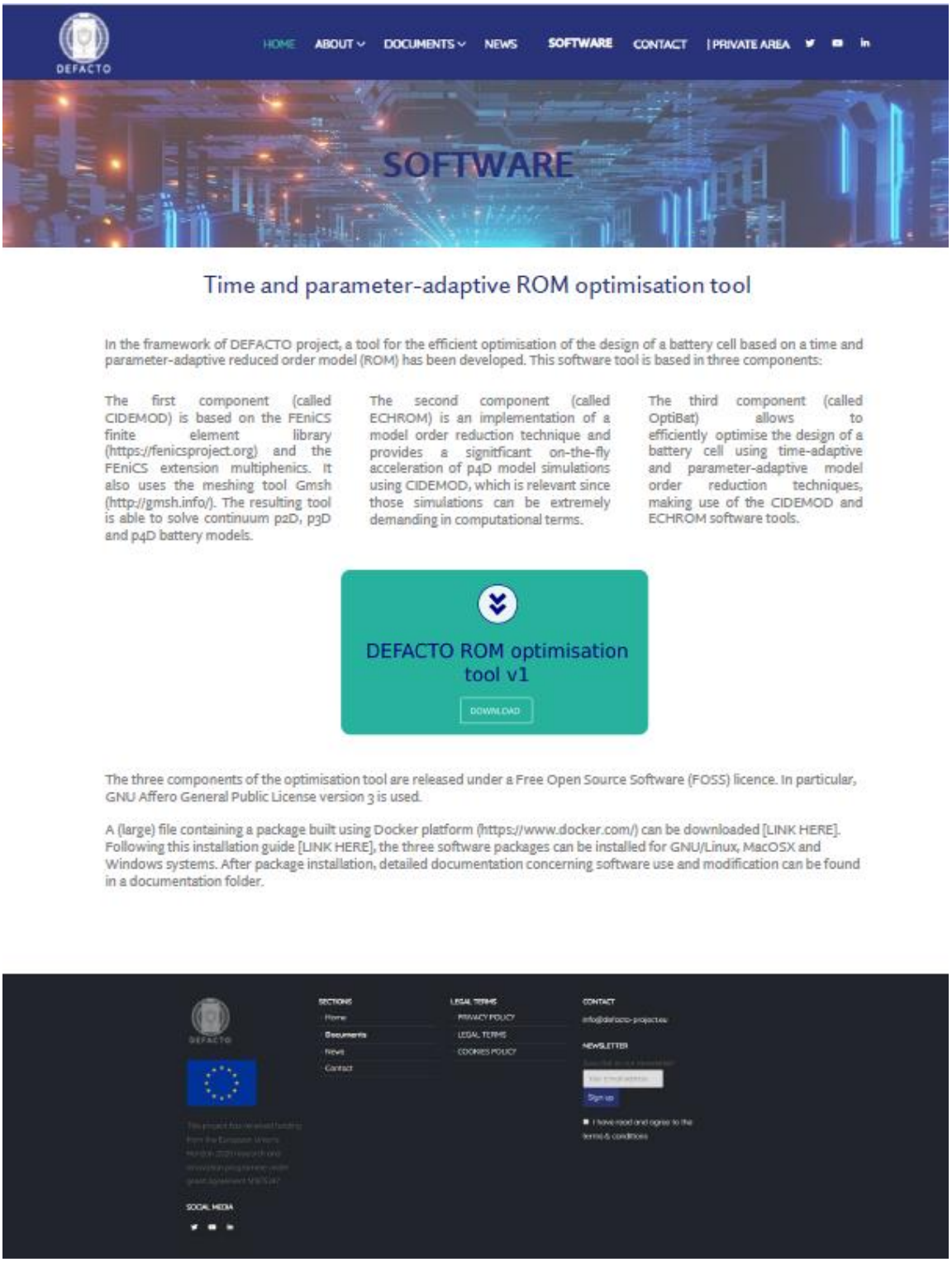

Figure 1. Screenshot of a preliminary design of the new section in DEFACTO website

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

8 Software installation

Software installation using Docker is very simple, since all the software tools are already *installed* inside the container used to distribute the software. In particular, developed codes, third party applications, libraries and configuration files (as well as a small operating system kernel) are bundled in the Docker container and only deployment of all these items on the final user machine is needed.

Software installation instructions are available at the project website (in a PDF file linked from software section, as shown above) and summarized here:

- Step 1: install Docker application following detailed instructions (according to the operating system in the computer, Docker Desktop or Docker Engine are recommended) at Docker website .
- Step 2: download Docker image file available at the Software section in DEFACTO project website (as shown in the previous document section).
- Step 3: start Docker application in the computer.
- Step 4: load downloaded image file to create a Docker container in the computer, following the instructions on the Docker website.

After completing these steps, all the distributed software is installed on the computer (the use of a Docker container ensures that software installation does not interact with any other program/library installation in the computer) and is ready to be executed. See or more information about using Docker containers. Some simple examples are included in a (local) folder in order to check software installation, as explained in the available installation instructions.

9 An application example case

In the distributed documentation (explained below) a collection of tutorials is planned, beginning with an example explained step by step. In order to make clear all aspects related to the software organization, a rather simple battery (consisting of a monolayer cell) is selected for this example case. More realistic batteries will be considered in subsequent examples.

Although this example is covered in detail in the documentation, a summary is included here to give a glimpse on the use of the developed software.

In order to use the p4D optimisation tool code for a specific battery configuration, the following data files are necessary:

- *params.json:* file containing the battery material and geometrical parameters. The name of this file can be changed by modifying the corresponding line in the *sim_dict.json* file.
- *sim_dict.json:* file containing the different simulation information for both complete and reduced order models. The name of this file can be changed by modifying the corresponding line in the *opt_dict.json* file.
- *adapt_dict.json:* file containing the needed parameters for the ROM adaptation strategy. The name of this file can be changed by giving it in the *opt_dict.json* file.
- *opt_dict.json:* file containing the needed parameters to perform the optimisation. The name of this file can be changed by modifying the corresponding line in the optimisation main script file.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

A detailed description of the first three files is presented in the echROM code documentation. Figure 2 shows an example of the opt_*dict.json* file. It contains the desired values for the parameters that must be provided to the software to perform the optimisation. A brief description of these parameters follows:

- *parameters*: list indicating the optimisation parameters to considered.
- *optimiser*: dictionary containing the needed information to perform the optimisation. It should indicate the following:
    - *algorithm*: SciPy optimisation algorithm to use.
    - *ftol* or *gtol*: tolerance for the optimisation algorithm convergence. See SciPy method options to find out which one is needed because it depends on the optimisation algorithm selected. For example, if *trust-const* method is considered, *gtol* must be provided. However, if *SLSQP* is selected use *ftol* instead.
    - *lower_bound*: list containing the lower bounds of all the considered optimisation parameters.
    - *upper_bound*: list containing the upper bounds of all the considered optimisation parameters.
    - *jac*: boolean to indicate to the optimiser if the derivatives of the objective function are provided (true). If false, the gradient will be estimated using 2-point finite difference estimation. Alternatively, the keywords "2-point", "3-point" or "cs" can be used to select the finite difference scheme.
    - *q_cell*: reference value of the battery cell capacity.
    - *penalty*: boolean indicating whether to add a penalty to the objective function (true) or not. This parameter is optional, its default value is false.
    - *callback*: boolean indicating whether to save the optimiser evolution (true) or not. This parameter is optional, its default value is false.
- *param_strategy*: dictionary containing the parameters to use in the parameter-adaptive strategy. It should indicate the following:
    - *usePS*: boolean to indicate whether to use the parameter-adaptative strategy (true) or not.
    - *projTol*: tolerance used to determine when the current information to construct the ROM is not enough.
    - *initialReductionFactor*: value by which *fomInitialTimeFrac* (provided in *adapt_dict.json* file) is reduced the first time the parameter-adaptive strategy is applied.
    - *minInitialTimeFrac*: minimum fraction of the total time each FOM must run each time it is called.
- *data*:
    - *sim_dict*: name of the *sim_dict.json* file to read.
    - *adapt_dict*: name of the *adapt_dict.json* file to read.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

```
{
    "parameters": ["porosity-a", "porosity-c"],
    "optimiser": {
        "algorithm": "SLSQP",
        "ftol":5e-4,
        "lower_bound": [0.1, 0.1],
        "upper_bound": [0.6, 0.6],
        "jac": true,
        "q_cell": 0.0022865394435258386,
        "penalty": false,
        "callback": true
    },
    "param_strategy": {
        "usePS": false,
        "projTol": 1e-1,
        "initialReductionFactor": 0.5,
        "minInitialTimeFrac": 0.025
    },
    "data": {
        "sim_dict":   "data/sim_dict.json",
        "adapt_dict": "data/adapt_dict.json"
    }
}
```

Figure 2. opt_dict.json file example

In addition, the needed data files and main program for a 1C discharge of a Gr-NMC cell optimisation are provided with the OptiBat package distribution. The characteristic and material parameters of this cell can be found in [8]. Note that the mesh used to spatially discretise the battery cell (see Figure 3) is relatively coarse in order to avoid high computational costs. Moreover, the height and width of the considered cell have been modified to values of an order comparable to the thickness. The configuration of this cell consists of negative electrode, separator, positive electrode.

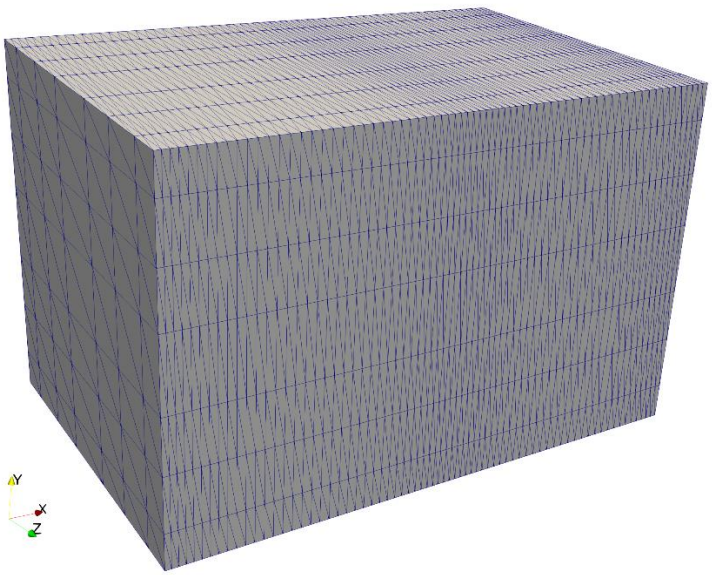

Figure 3. Battery cell mesh

The main program script, after reading the *opt_dict.json* input file described above, creates an instance of the *Optimiser* class. It reads the parameters provided in all the inputs files and creates a callback to the selected objective function. Finally, the *solve* method of the *Optimiser* class is called to perform the optimisation for the given initial set of parameters. Figure 4 shows a screenshot of this script.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

```python
import os
os.environ['OMP_NUM_THREADS'] = '1'

import sys
sys.path.insert(0, 'src')

import json

from OptiBat.opt_classes import Optimiser

# Load input json file as dictionary
with open('data/opt_dict.json') as json_file:
    opt_dict = json.load(json_file)

# Create an Optimiser instance
optim = Optimiser(opt_dict, 'max_energy')

# Solve optimisation problem
opt_p = optim.solve([0.25, 0.335])
```

Figure 4. Python code screenshot showing the optimisation problem resolution

Figure 5 shows the evolution of the objective function during the optimisation process obtained using the time- and parameter-adaptive p4D optimisation tool for the described cell configuration. The software uses adimensional values for the optimisation parameters and for the objective function, in order to avoid problems derived from the treatment of different scales. Thus, the values of the delivered energy presented in Figure 5 are always less than one. Note that within a few iterations, the optimiser reaches a set of the optimisation parameters for which the value of the objective function is closest to its maximum. It is not worth continuing to iterate (which increases the computational cost of the optimisation) because the improvement in the delivered energy achieved in the next iterations will be of the order of the numerical errors derived from the resolution of the models. A more detailed explanation of this optimisation example can be found in the corresponding code tutorial.

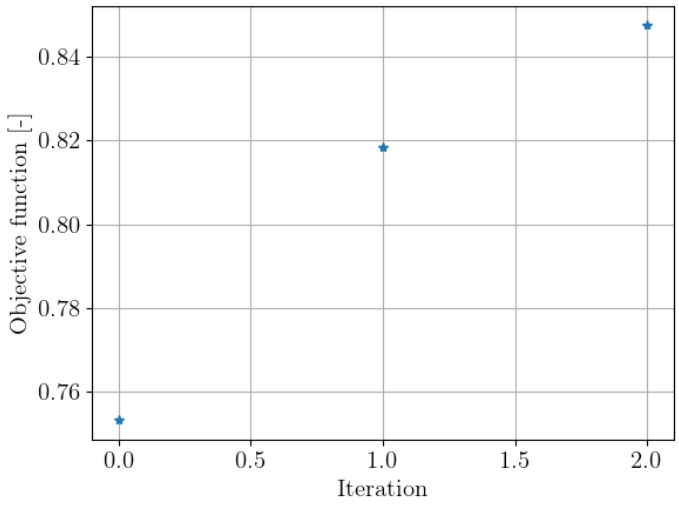

Figure 5. Objective function value along the optimisation process.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

10 Software documentation

Documentation distributed with the software is organized in three categories:

1. Theory manuals, intended to explain underlying algorithms (especially concerning model order reduction techniques) and providing guidelines on the efficient use of the software
2. Programming manuals, explaining code internal organization, use of internal and external libraries and programming tips
3. Tutorials, based on detailed examples intended to cover all the aspects of the software use

This documentation will be enriched and updated throughout the DEFACTO project execution, as new software features are incorporated and feedback from software users is obtained.

Concerning the first category, a theory manual is distributed with the first release of the code. This manual explains the ideas behind time-adaptive model order reduction techniques used in the software development, and how these ideas have been implemented in the software. This theory manual will be completed soon with a scientific paper exploring the efficient selection of the algorithm parameters and thus providing practical guidelines on the parameter selection.

Programming documentation (more oriented to developers, including those users aiming to modify the software to cover specific functionalities) includes a document detailing the main aspects concerning the software organization. Regular software users are expected to interact only with a Python script acting as a main program and able to (easily) configure all algorithm parameters. Details of the algorithm implementation instead are encapsulated in subprograms. Accurate and extensive comments have been included in all the source files. Documentation on Python subprograms has been collected using the Sphinx tool (https://www.sphinx-doc.org) thus providing a very convenient interface to access software documentation. Figure 6 shows an example of this interface.

The last category corresponds to software tutorials. The first tutorials focus on software fundamentals and selection of most relevant algorithm parameters. More advanced tutorials (dealing with more realistic battery models as well as more sophisticated algorithms) are being developed. As suggested above, this section will be progressively enlarged as feedback from users is obtained.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

```python
##########################################
###            MASS MATRIX             ###
##########################################
def M_A_gen(coef, psi_i, psi_j, dofs_index, volum):
    """
    Assemble the reduced order model general mass matrix:

    .. math::
        M_{ji} = \\int_{\Omega_d} c \\psi_i \\psi_j d\\Omega_d

    being:\n
        * :math:`c` a constant coefficient.\n
        * :math:`\\psi_i` the :math:`i`-th mode associated to the involved trial function.\n
        * :math:`\\psi_j` the :math:`j`-th mode associated to the involved test function.\n
        * :math:`\Omega_d` a problem subdomain.\n

    :param coef: value of the coefficient :math:`c`.
    :type coef: float

    :param psi_i: Modes for the trial function.
    :type psi_i: numpy.ndarray

    :param psi_j: Modes for the test function.
    :type psi_j: numpy.ndarray

    :param dofs_index: P1 dofs indices for each finite element.
    :type dofs_index: numpy.ndarray

    :param volum: Volume of the finite element of the subdomain :math:`\Omega_d` .
    :type volum: numpy.ndarray

    :return: Assembled reduced order model general mass matrix.
    :rtype: numpy.ndarray
    """

    # Define quadrature points and weights in the reference tetrahedral
    quad = quadrature.Quadrature('tetra', 2)
```

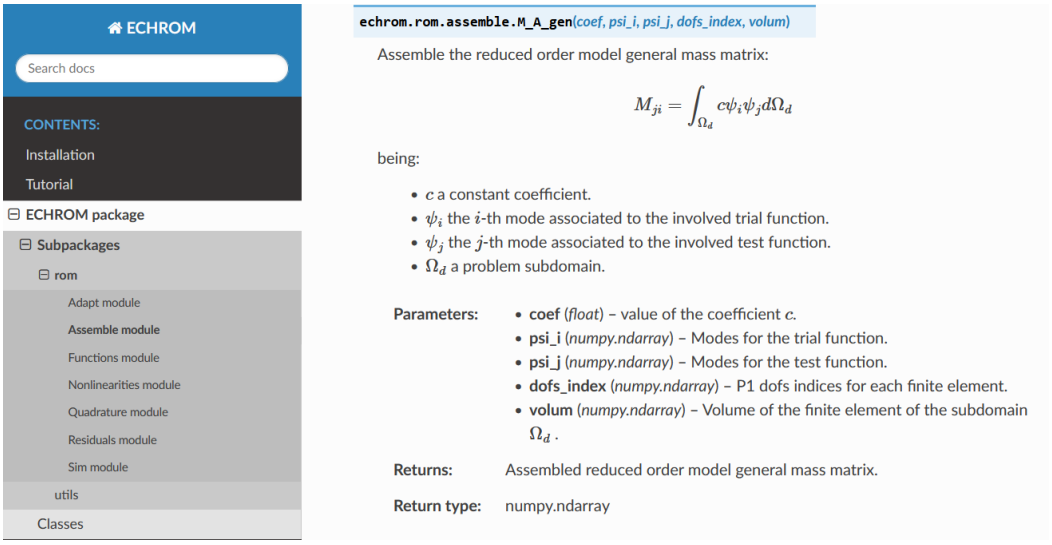

Figure 6. Example of a commented code (top image) and its corresponding documentation interface automatically generated by Sphinx tool (bottom image).

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

11 Conclusions

As explained in the current D6.2, a first release of the time- and parameter-adaptive DEFACTO reduced p4D optimisation tool (developed in WP6 task T6.4 and based on the time-adaptive DEFACTO reduced p4D software tool in WP6 task T6.1) under a Free Open-Source Software (FOSS) license has been accomplished.

In this document, an explanation of the most important aspects of this software release (which constitutes the D6.2 deliverable itself) has been presented. In particular, (a) the main features of the software tool have been described, (b) selection of a suitable FOSS license has been justified, (c) modification of the project website to host the software release has been illustrated, (d) software distribution and installation has been briefly described, (e) a short overview of software use through a simple example has been presented, and (f) distributed documentation has been discussed.

Finally, as planned in the DEFACTO project proposal, software capabilities will be expanded throughout the project execution. Also, the documentation section will be expanded as feedback from first users begins to be received. All these updates will be included in future software releases.

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247

12 Bibliography

[1] Alnæs, Martin et al., "The FEniCS project version 1.5," *Archive of Numerical Software,* vol. 3, no. 100, 2015.

[2] "multiphenics - easy prototypinf of multiphysics problem in fenics," [Online]. Available: https://mathlab.sissa.it/multiphenics.

[3] C. Geuzaine, J. Remacle, "Gmsh," 2020. [Online]. Available: http://http://gmsh.info/.

[4] M.L. Rapún, F. Terragni, J.M. Vega, "Adaptive POD-based low-dimensional modelling supported by Residual Estimates," *Int. J. Numerical Methods in Engineering,* vol. 104, no. 9, pp. 844-868, 2015.

[5] S. LeClainche, F. Varas, J.M. Vega, "Accelerating Oil Reservoir Simulations using POD on the fly," *Int. J. Numerical Methods in Engineering,* vol. 110, no. 1, pp. 79-100, 2017.

[6] M. Doyle, T.F. Fuller, J. Newman, "Modeling of Galvanostatic Charge and DIscharge of the Lithium/Polymer/Insertion Cell," *J. Electrochem. Soc.,* vol. 140, no. 6, pp. 1526-1533, 1993.

[7] M. Safari, M. Morcrette, A. Teyssot, C. Delacourt, "Multimodal Physics-Based Aging Model for Life Prediction of Li-ion Batterie," *J. Electrochem. Soc.,* vol. 156, no. 3, p. A145, 2009.

[8] C. Chen, F.B. Planella, K. O'regan, D. Gastol, W.D. Widanage, E. Kendrick, "Development of experimental techniques for parameterization of multi-scale lithium-ion battery models," *J. Electrochem. Soc.,* vol. 167, no. 8, p. 080534, 2020.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 875247