DEFACTO - battery DEsign and manuFACTuring **Optimisation through multiphysic modelling**

D.6.1

Date: 31.12.2021

This document is a description of the DEFACTO D6.1 deliverable (contract no. 875247 coordinated by CIDETEC). This document explains the details concerning the release of the time-adaptive DEFACTO reduced p4D software tool (based on a p4D modelling tool, also developed in the framework of the DEFACTO project) under a Free Open-Source Software (FOSS) license, which constitutes the deliverable D6.1 itself. In the document, both the present capabilities of this tool and their extension in the planed 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	*D6.1*		
Title	*Release of the time-adaptive DEFACTO reduced p4D tool based on DEFACTO p4D model under FOSS*		
Work Package	*WP6 – Optimisation and sensitivity analysis*		
Dissemination level	Public	**Nature**	*Other (Software)*
Due date (M)	M24	**Submission date (M)**	*M24*
Deliverable responsible	UPM	Contact persons	*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)	Elixabete Ayerbe	CID	Partner	eayerbe@cidetec.es
	Raúl Ciria	CID	Partner	rciria@cidetec.es
	Clara Ganuza	CID	Partner	cganuza@cidetec.es
	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
	Fernando Varas	UPM	Partner	fernando.varas@upm.es
Reviewer(s)	Jean-Pierre Cazeaux	ESI	WP6 leader	*jean-pierre.cazeaux@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
10.12.2021	1.0	UPM	Initial draft
10.12.2021	1.1	Jean-Pierre Cazeaux	Revision by WP leader
17.12.2021	2.0	UPM	Implementation of revision comments
20.12.2021	2.1	CID	Revision by coordinator
10.01.2022	5.0	UPM	Implementation of revision comments
11.01.2022	5.1	CID	Revision by coordinator
11.01.2022	6.0	UPM	Implementation of revision comments
04.03.2022	FV	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

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

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
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

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 packages 5 and 6, an adaptive reduced order model to accelerate p4D model simulations is being developed. As stated in DEFACTO grant agreement, the software implementing this reduced p4D model will be released as Free Open-Source Software (FOSS) license. Deliverable D6.1 corresponds precisely to the first release of this software.

This software is based on two 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.

In this initial release, an electrochemical model is considered. In later software releases, as also described in the grant agreement, this model will be completed with mechanical and ageing models.

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 (titled 'Software') is being added to the website. To make software installation easier, both tools (cideMOD and ECHROM) 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 container 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 D6.1 corresponds to the release of a software tool able to efficiently solve continuum pseudo four-dimensional models using time-adaptive model order reduction techniques. This document contains a description of the D6.1 deliverable and not the deliverable itself that, as described below, is available at the DEFACTO project website. According to the DEFACTO proposal this software is released under a Free Open-Source Software (FOSS) license.

The released software is internally organized in two components:

- The first one, called cideMOD, is being developed in the framework of the 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 () and an extension of this library called multiphenics (developed with the support of the European Research Council, ERC). A meshing tool Gmsh (https://gmsh.info/) 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 [1] [2], 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).

These software components will be continuously developed along 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 is a generalization of the Doyle-Fuller-Newman [3] 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 the 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 simplified interface to run automatically arbitrary testing conditions, cycling protocols and usage profiles.

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

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:
  - Taking in 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) [4].

5.2 Features of ECHROM component

As previously described, ECHROM component goal is the acceleration of (computationally demanding) p4D simulations with cideMOD software using adaptive model order reduction techniques

According to this goal, evolution of ECHROM component will closely follow cideMOD component development along DEFACTO project execution. At this stage, only an electrochemical model can be accelerated (since continuum mechanical and ageing models are still under development and only an electrochemical ageing model related to SEI growth is included in cideMOD component). A thermal model, already available in p4D, will be included in ECHROM acceleration tool very soon. Upon complete programming and validation of continuum mechanical and ageing/damage models in cideMOD, they will be included in ECHROM component too.

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, a minimum interaction concerning algorithm parameter selection is required if the user decides to accept default parameter selection. ECHROM component can be used providing exclusively the same JSON input file needed to simulate the battery using cideMOD library. 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).

Figure 1 In the next figure a screenshot of this script is shown.

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
####### FOM preprocess SKIPPED CODE  #######
# Solve first FOM (initial time set to 0 by default)
fom_status = problem.solve_ie(min_step=min_step, i_app=I_app,
                              t_f=int(t_sim_end*fomInitialTimeFrac),
                              store_delay=-1, adaptive=False, triggers=[v_min])

# Store initial FOM results in adaptive model results dict
adaptResDict = problem.fom2rom['results'].copy()

####### ROM preprocess SKIPPED CODE  #######
# Set final time solve by FOM as initial time for ROM
t_ini = problem.fom2rom['results']['time'][-1]
# Set final time to solved with ROM
t_end = min(t_sim_end, t_ini+romInitialTimeFrac*t_sim_end)

# Solve first ROM
rom_results, rom_status = sim.solve(rom_dict, cell_dict, rom_system,
                                    t_ini, t_end, tadapt_step, verbose=False)

# Store initial ROM results in adaptive model results dict
adaptResDict = concatenate_results(adaptResDict, rom_results)

# Loop in the maximum number of adaptations allowed
for nt in range(tadapt_loop_maxiter):

    ####### FOM preprocess SKIPPED CODE  #######
    last_timeROM = rom_results['time'][-1] # Last time solved by the previous ROM
    # Set last time step to solve with the new FOM
    t_end = min(t_sim_end, last_timeROM+fomInitialTimeFrac*fomTimeReduction*t_sim_end)

    # Solve FOM (initial time modified in the preprocess)
    fom_status = problem.solve_ie(min_step=min_step, i_app=I_app, t_f=t_end,
                                  store_delay=-1, adaptive=False, triggers=[v_min])

    # Store FOM results in adaptive model results dict
    adaptResDict = concatenate_results(adaptResDict, problem.fom2rom['results'])

    # Check if the cut-off voltage or the final time are reached
    if fom_status == 1 or romAdaptResDict['time'][-1] >= t_sim_end:
        break

    ####### ROM preprocess SKIPPED CODE  #######
    # Set final time solve by FOM as initial time for ROM
    t_ini = problem.fom2rom['results']['time'][-1]
    # Set final time to solved with ROM
    t_end = min(t_sim_end, t_ini+romTimeFrac*t_sim_end)

    # Solve ROM
    rom_results, rom_status = sim.solve(rom_dict, cell_dict, rom_system, t_ini,
                                        t_end, tadapt_step, verbose=False)

    # Store ROM results in adaptive model results dict
    adaptResDict  = concatenate_results(adaptResDict, rom_results)

    # Check if the cut-off voltage or the final time are reached
    if rom_status == 1 or romAdaptResDict['time'][-1] >= t_sim_end:
        break
```

Figure 1. Python code screenshot showing (adaptive) model order reduction implementation

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

Concerning output file formats, since reduced order model solution conversion 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, and 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, 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).

In order to obtain additional acceleration, information from a simulation database (if available) can be introduced in the adaptive model order reduction technique. This feature is being implemented in the software and it will be available in the next release.

6 Free Open-Source Software license

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 make possible this software can benefit from contributions by other users.

Concerning the selection of a particular FOSS license to release developed software tools, 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 viral license was selected
- 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) and Gmsh (GPLv2). Finally, the same FOSS license was used for both software tools, cideMOD and ECHROM, to make easier 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.*

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 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 possible to modify the code if this is needed in any future software update.

7 Distribution through the project website

7.1 Software section in DEFACTO project website

Concerning software distribution, it was decided to use 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, a new section (titled 'Software') was opened in DEFACTO project. This section will host software tools developed in the project (in the framework of work package WP6, release of a commercial software tool is also planned). Figure 2 shows a screenshot of the presentation of the reduced p4D model software tool in the new section of the project website.

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

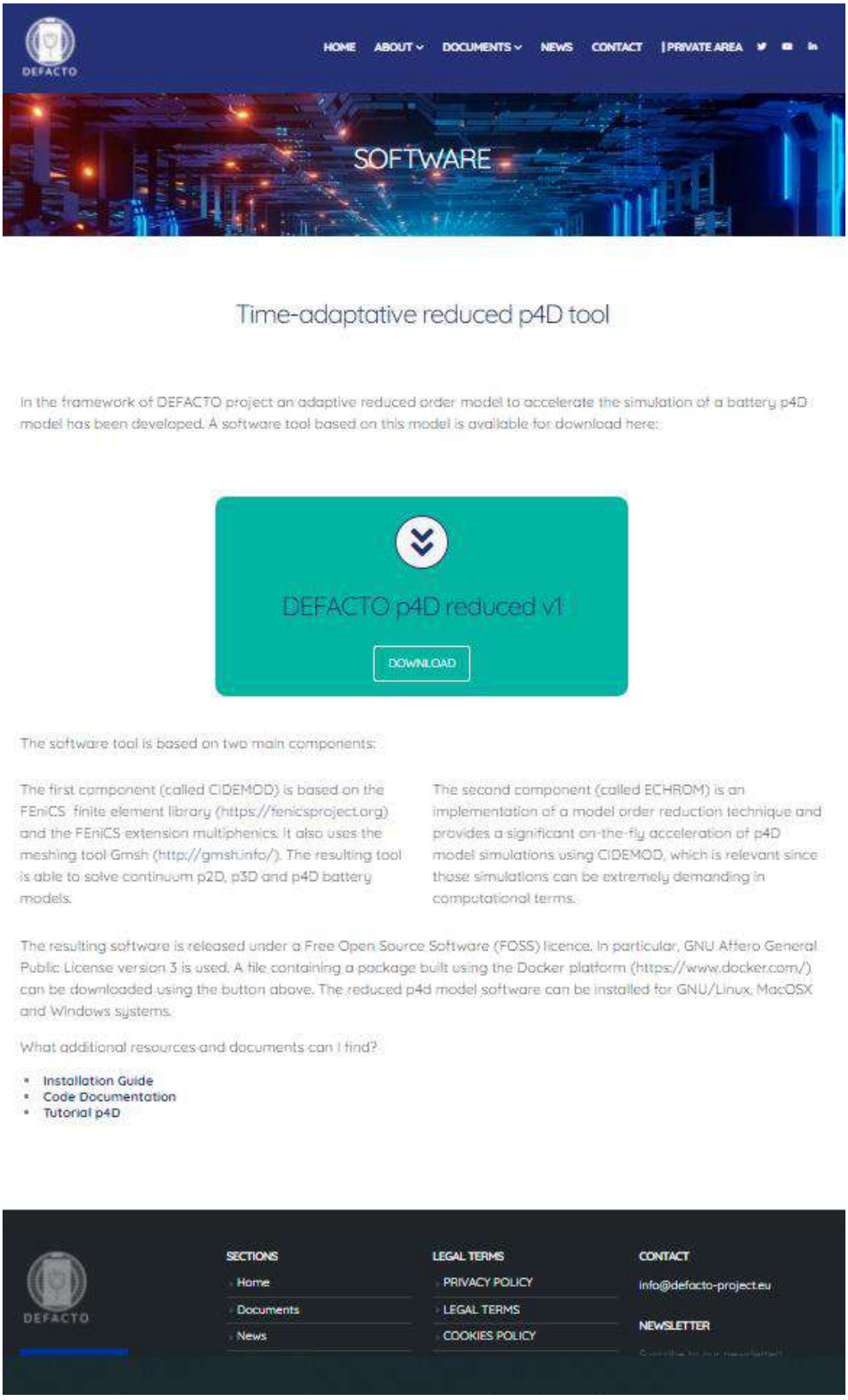

Figure 2. Screenshot of the new section in DEFACTO website

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

7.2 Software packaging tool

In order to distribute the developed software tools, it was decided to make software distribution as easy as possible, in order to promote a wider dissemination. Thus, instead of releasing exclusively DEFACTO p4D reduced software (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(https://www.docker.com/) 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 available at DEFACTO website can thus be used to run reduced p4D 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 continuum p4D simulation itself will usually require expensive computations. Experimented users requiring intensive execution of reduced p4D 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).

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 rather small operating system kernel) are bundled in Docker container and only deployment of all these items in 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 container 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 instructions at Docker website

After completing these steps, all the distributed software is installed in 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 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.

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

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 single cell) is selected for this example case. More realistic batteries are 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 time-adaptive p4D reduced order model 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 giving it in the sim_dict.json file. A detailed description of its format is presented in the code documentation.
- *sim_dict.json:* file containing the different simulation information for both complete and reduced order models. A detailed description of its format is also presented in the code documentation.
- *adapt_dict.json:* file containing the needed parameters for the ROM adaptation strategy.

If the user does not want to modify the model order reduction strategy provided by the default parameters, only params.json and sim_dict.json files must be prepared and the main program can be launched without any modification. If the user wants to modify this strategy, file *adapt_dict.json* must be edited. Figure 7 shows an example of *adapt_dict.json* file. It contains the desired values for the parameters that must be provided to the time-adaptive model order reduction technique. A briefly description of these parameters follows:

- *fomInitialTimeFrac*: fraction of the total simulation time, at the beginning of the simulation, the problem must be solved using the full order model by cideMOD.
- *fomTimeReduction*: fraction of the simulation time using cideMOD at the beginning of the simulation (determined by the previous parameter) that the problem must be solved using the full order model (provided by cideMOD) each time it is called again.
- *romInitialTimeFrac:* maximum fraction of the total simulation time the problem should be solved using the first reduced order model.
- *romTimeFrac*: maximum fraction of the total simulation time the problem should be solved using any reduced order model.
- *romSVDtolAdapt*: tolerance used in the SVD decomposition in order to determine the number of modes to represent each field at each reduced order model update event.

```
{
    "fomInitialTimeFrac": 0.05,
    "fomTimeReduction": 0.25,
    "romInitialTimeFrac": 0.1,
    "romTimeFrac": 0.95,
    "romSVDtolAdapt": 1e-5
}
```

Figure 3. adapt_dict.json file example

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 addition, the needed data files and main program for a 1C discharge of a graphite-NMC cell are provided with the ECHROM package distribution. The characteristic and material parameters of this cell can be found in [5]. The configuration of this cell consists in negative current collector, negative electrode, separator, positive electrode and positive current collector. On the other hand, a detailed description of the main program implementation is available in the code documentation. However, in the present document only the general idea will be shown.

The main program script, after reading all the inputs files described above, defines the full order model (FOM) resolution options as a p4D and a non-dimensional model and solve a few time steps to generate a basis for the first considered reduced order model. After the FOM resolution, the first ROM is constructed and the final time step solution of the FOM is given to the ROM as initial condition. Also, a first ROM is solved as is shown in Figure 4.

Subsequently, if the given final time and the given cut-off voltage are not still reached, the solution of the last time step solved by the first ROM is provided as initial condition to a second FOM which will be solved, as shown in Figure 5. Again, once the second FOM finish the time resolution, the last time step solution is given to another ROM which will be constructed and solved. This process will continue until either the given final time or the cut-off voltage is reached. It should be noted that if any of the FOM and ROM models verify any of the previously described stop criteria, the simulation ends, and the results obtained are returned to the user.

```
############################ ROM #############################
# Build ROM dictionaries
cell_dict, rom_dict = pre.build_rom_dict(dic_cell, dic_sim, problem.fom2rom, data_path)

# Store initial FOM results in adaptive model results dict and in fomFields object
fomFields = classes.CellFields()
fomFields.loadFomData(problem.fom2rom['results'], rom_dict)

# Build non-linearities from FOM solution to perform SVD in the ROM
pre.build_nl_fields(fomFields, rom_dict['romNonLinearFields'], cell_dict, rom_dict)

# Use fjLi to describe jLi
fomFields.getField('jLi').data = fomFields.getField('fjLi').data.copy()

# Perfom SVDs and build ROM system
rom_system = sim.build_ini(rom_dict, cell_dict, fomFields, tol=rom_dict['romSVDtol'])

# Set ROM initial conditions
for key in problem.fom2rom['results']:
    if key != 'time' and key != 'voltage':
        rom_system.getEq(key).init = problem.fom2rom['results'][key][:, -1]

# Solve initial ROM
rom_results, rom_status = sim.solve(rom_dict, cell_dict, rom_system,
                            problem.fom2rom['results']['time'][-1],
                            min(tadapt_final, problem.fom2rom['results']['time'][-1]+
                                tadapt_rom_ini_f*tadapt_final),
                            tadapt_step, verbose=False)
```

Figure 4. First ROM construction and resolution implementation.

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

```
############################ FOM ############################
# Handle FOM initial conditions
fom_init = dict()
for key in rom_results:
    if key != 'time' and key != 'voltage':
        fom_init[key] = rom_results[key][:, -1].copy()

# Solve FOM
problem.reset2rom(rom_results['time'][-1], fom_init)
fom_status = problem.solve_ie(min_step=min_step, i_app=I_app,
                              t_f=min(tadapt_final, rom_results['time'][-1]+
                                      tadapt_fom_f*tadapt_fom_red_f*tadapt_final),
                              store_delay=-1, adaptive=False, triggers=[v_min])
```

Figure 5. FOM resolution after a ROM calculation.

Figure 6 shows the results obtained with the time-adaptive p4D reduced order model for the described cell configuration. The figure at the left shows the voltage obtained with the reduced order model and the figure at the right a comparison within FOM and ROM voltages. As can be seen in these figures, both model returns quite similar results.

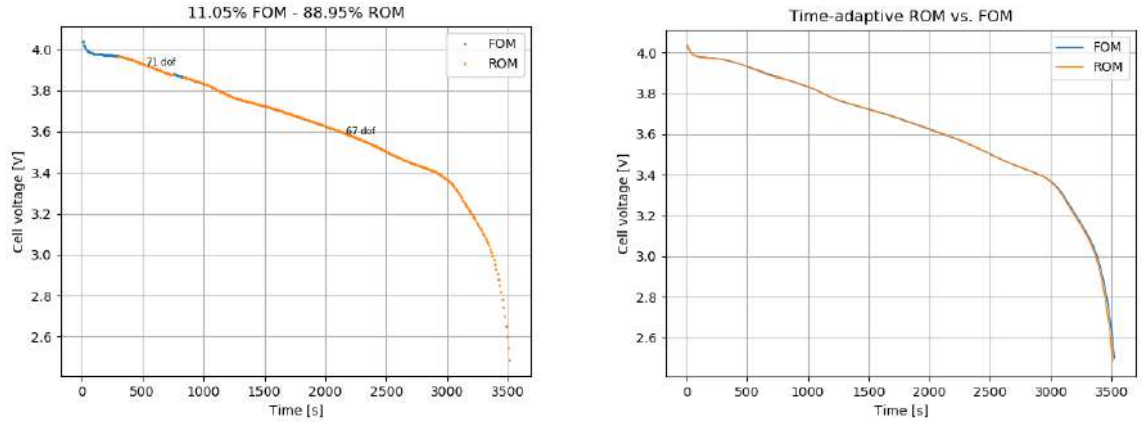

Figure 6. Voltage ROM results (left) and voltage comparison between ROM and FOM (right).

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 along 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 Sphinx tool (https://www.sphinx-doc.org) thus providing a very convenient interface to access software documentation. Figure 7 shows an example of this interface.

Last category corresponds to software tutorials. First tutorials focus on software fundamentals and selection of most relevant (model order reduction) 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.

Last category corresponds to software tutorials. First tutorials focus on software fundamentals and selection of most relevant (model order reduction) 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.

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)
```

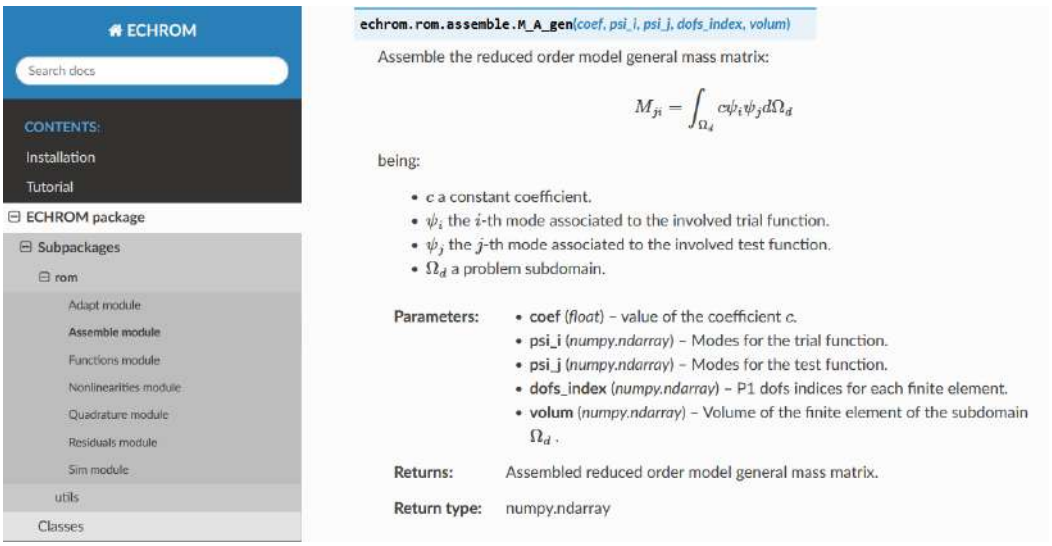

Figure 7. Example of a commented code (top image) and its corresponding documentation interface automatically generated by Sphinx tool (bottom image).

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 this document, deliverable D6.1 goals have been reached. Namely, a first release of the time-adaptive DEFACTO reduced p4D software tool (developed in WP6 task T6.1 and based on DEFACTO p4D model developed in WP5 task T5.5) 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 D6.1 deliverable itself) has been presented. In particular, (a) main features of the software tool have been commented, (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 DEFACTO proposal, software capabilities will be expanded along the project execution. Also, documentation section will be enlarged as feedback from first users begins to be received. All these updates will be included in future software releases.

12 Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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