Electric Motor Optimization

Yann Debray – Scilab – ESI Group

Abstract

With today's trend for electric car, every car maker must go back to specifications and systems engineering to replace basic powertrain functions. In order to make this electric car-paradigm a reality, suppliers have therefore to provide car manufacturers with optimal design taking into account new design tradeoffs such as autonomy and power to be delivered.

In this context, ESI has been working with both manufacturers and suppliers of the automotive domain on optimizing the design of electric motors (and associated batteries). Considering that such design requires many geometric changes (number of poles rotor/stator, number of coil's turns, number of phases, …), **Finite Element Analysis** (FEA) prove to be too expensive in terms of computing power for large design space exploration on a dynamic model (each simulation taking several hours).

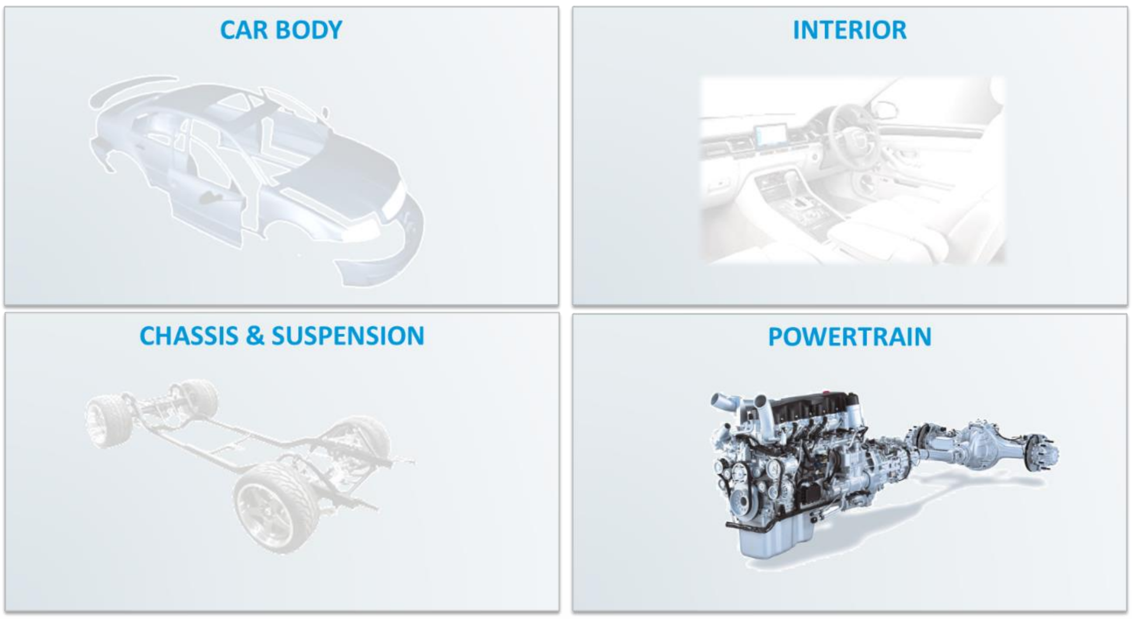
Following clients' needs, ESI solutions are as followed: First, simulate multi-physics static finite element models. Second, integrate reduced models in dynamic system models with SimulationX. Second, integrate the solution with Scilab through the **Functional Mock-up Interface** (FMI) and giving access to the results via a tailored application.

Model reduction and **optimization** techniques can then be leveraged in order to consider the changes in the geometry and the design parameters. Future developments will be about interfacing additional external tools & libraries with Scilab (like Octave for Matlab compatibility) to serve as an integration hub and deployment platform for both academics & industries.

In this paper, we display the benefits of an approach around Model-based Design and Optimization, coming from the experience of the Scilab & SimulationX teams with ESI Group.
SimulationX is a Model-Based Design environment relying on the open language Modelica, for the simulation of mechatronic systems.
Scilab is an open-source software for numerical computing and scientific programming. Every month, 100 000 engineers and scientists download Scilab around the world

1. Electric motor modeling

System Modeling enables full vehicle simulation at different levels of fidelity. In the following graph, we represent the architecture of a vehicle, around the main engineering department (body, interior, chassis & powertrain).

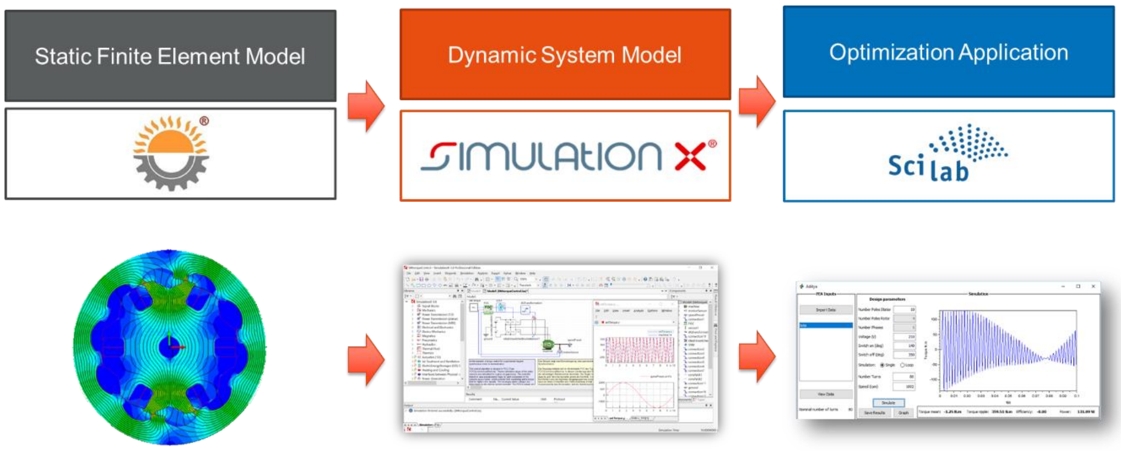


In this paper, we will address a central topic in the engineering of electric vehicle:
The design and optimization of an electric motor.
In order to define the requirement of such a system we need to cascade the expected performances of an electric vehicle to the different sub-systems, model and analyze the interactions, and freeze the design at some point of the vehicle program.

1.1. Modeling workflow

The following workflow represents the 3 main steps of the modeling and optimization.
Existing 2D finite element models of previous generation of motors are leveraged in order to optimize the main elements of the design:

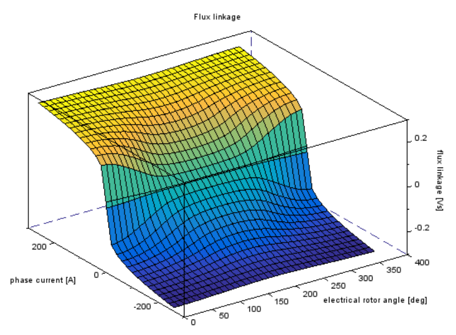
- Number of Poles of the motor (Rotor and Stator),
- Number of Turns of the coil
- Operating Voltage
- Operating Speed

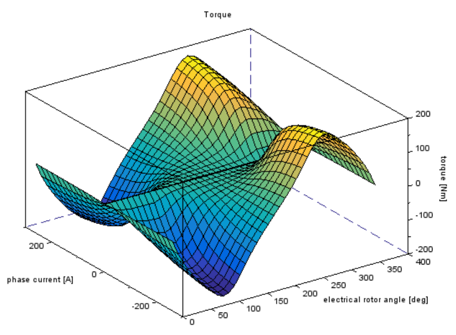


1.2. Static Finite Element Model

The typical Finite Element Model (FEM) would be simulated in the **static** domain and would take around **3 hours** (static meaning for every angular position of the rotor). A simulation in the **dynamic** domain will take about **30 hours**. Further there could be about **3-4 continuous and discrete variables** that participate in characterizing the problem. This would need a huge number of runs to come to grips with the behavior. This coupled with multi-objective optimization would lead to a prohibitive amount of computing time, that practically renders optimization on a dynamic finite element model unviable.

What is considered here is to use a system model in order to perform multi-objective optimization on the design, based on calibration values from the finite element model. The values that are provided as output of this models are for flux linkage and torque (depending on angle in degrees and current):





1.3. Dynamic system model

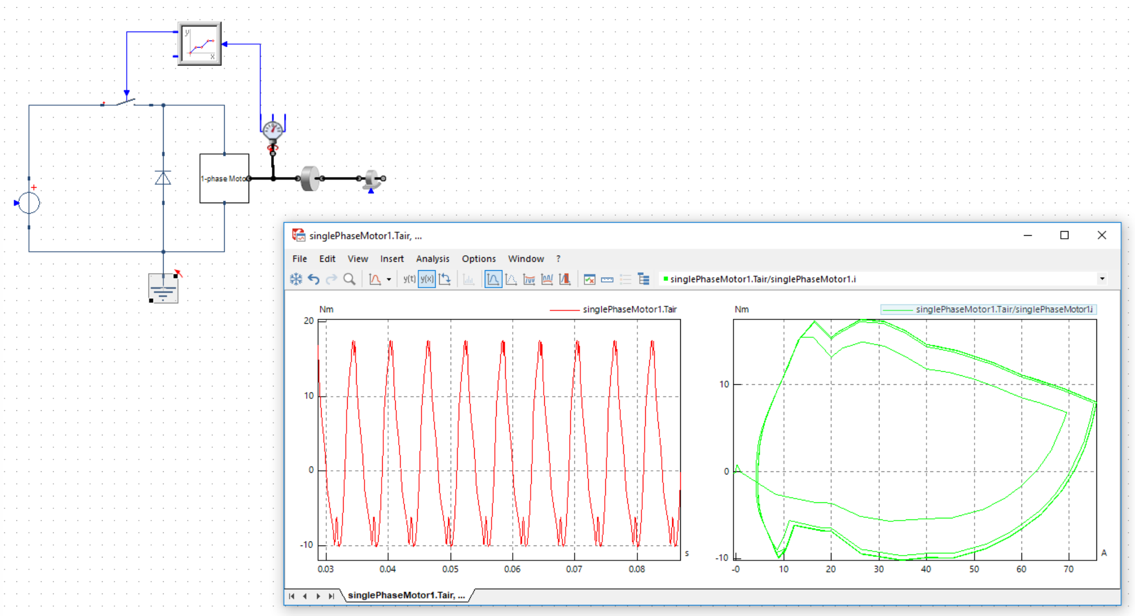
The modeling of a dynamic system was performed in the following steps:

- Single phase motor model development
- Three phase motor model development
- Modelling voltage supply and power switches (switching circuit)
- Correlation with test data (not covered un this paper)

1.3.1. Single phase motor

In order to get started with a simplified model, we engaged with the modeling of a single-phase motor:

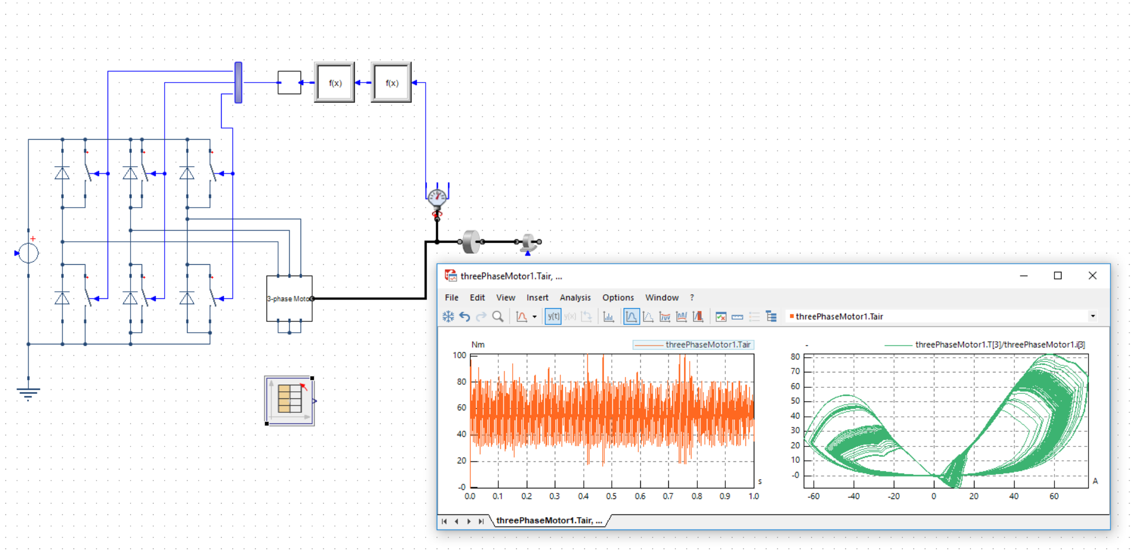
- Simple model
- Single phase with switching
- Imports Torque and Lambda data
- Dynamically plot values of
    - Current
    - Torque
    - Power


1-phase Motor
singlePhaseMotor1.Tair, ...
File Edit View Insert Analysis Options Window ?
singlePhaseMotor1.Tair/singlePhaseMotor1.i
Nm
singlePhaseMotor1.Tair
20
10
0
-10
0.03
0.04
0.05
0.06
0.07
0.08
s
Nm
singlePhaseMotor1.Tair/singlePhaseMotor1.i
10
0
-10
-0
10
20
30
40
50
60
70
A
singlePhaseMotor1.Tair, ...

1.3.2. Three Phase Motor Model – Version 1

Once this simple model has shown some of the basic performance indicators that we were interested in, we switched to the modeling of a three phase motor:

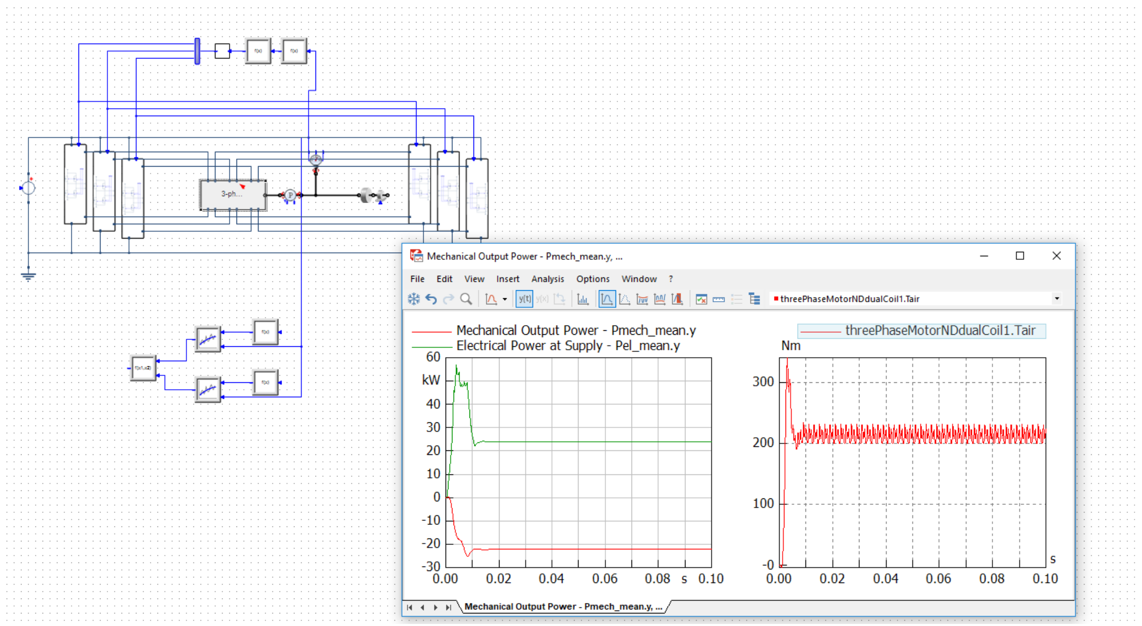
- Three phase star connected motor
- Rotor position based switching
- Simple three phase inverter switching circuit topology
- Imports Torque and Lambda data
- Fixed no. of poles
- Dynamically plot (vs time or any desired quantity)
    - Current
    - Air gap torque
    - Power



f(x)
f(x)
3-phase Motor
threePhaseMotor1.Tair, ...
File Edit View Insert Analysis Options Window ?
threePhaseMotor1.Tair
Nm
threePhaseMotor1.Tair
threePhaseMotor1.T[3]/threePhaseMotor1.i[3]
s
A
threePhaseMotor1.Tair, ...

1.3.3. Three Phase Motor Model – Version 2 (Voltage Supply and Power Switches)

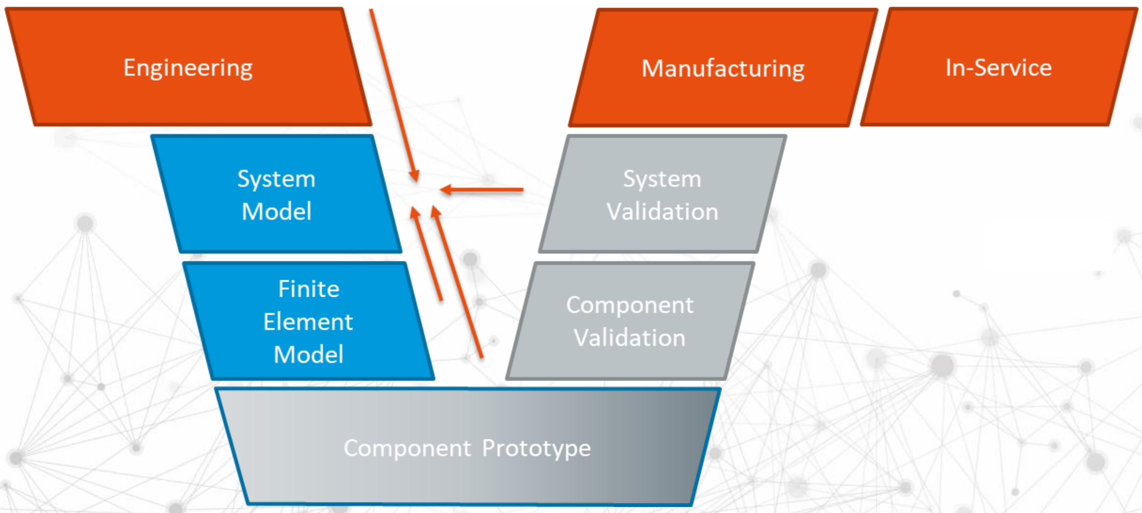
The three phases are now independent, and each phase has two coils controlled by simultaneously switched H Bridge. We also implement a variable no. of poles and turns.


3-ph...
Mechanical Output Power - Pmech_mean.y, ...
File Edit View Insert Analysis Options Window ?
threePhaseMotorNDdualCoil1.Tair
Mechanical Output Power - Pmech_mean.y
Electrical Power at Supply - Pel_mean.y
kW
60
40
30
20
10
0
-10
-20
-30
0.00 0.02 0.04 0.06 0.08 s 0.10
threePhaseMotorNDdualCoil1.Tair
Nm
300
200
100
-0
0.00 0.02 0.04 0.06 0.08 0.10
s
Mechanical Output Power - Pmech_mean.y, ...

2. Import the model with the Functional Mock-up Interface

2.1. Introduction of system in the modeling workflow

It might seem peculiar that in the scenario of this paper and project, we proceed from a finite element model to a system model, as represented in the modeling workflow section. Typically, the process we observe around model-based design and model-based systems engineering introduces a V cycle, providing a functional system layer on top of the component layer.



In large automotive manufacturers, the motor department are then provided by functional requirements from the systems engineers, and component models are developed based on this to fulfill the technical specification.

2.2. Functional Mock-up Interface

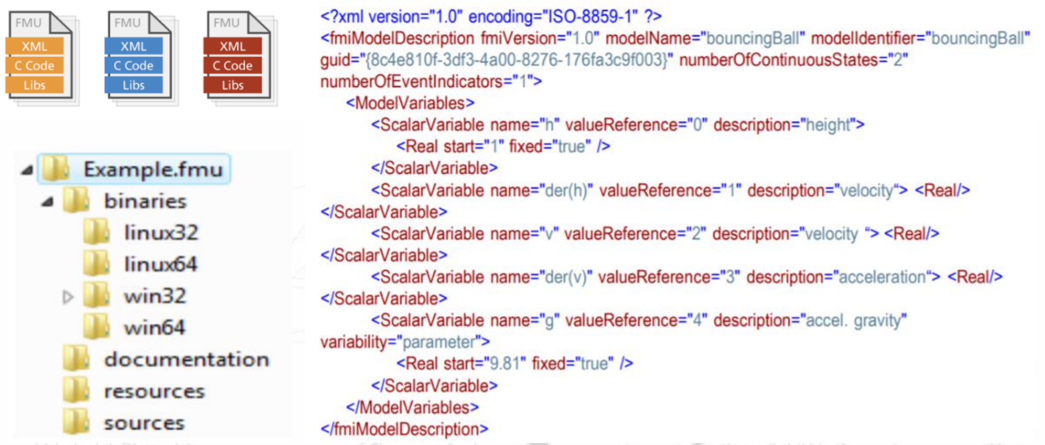
The Functional Mock-up Interface standard has been initiated and coordinated by Daimler and has been developed within the MODELISAR consortium from 2008 to 2011, alongside the MODELICA open source language.
The main benefits of this standard are to:

- Free the model unit (FMU) from license restrictions
- Separate the model authoring tool from model execution tools
- Deploy from few simulation specialists to designers, domain specialists, control engineers

In terms of implementation, two modes are available:

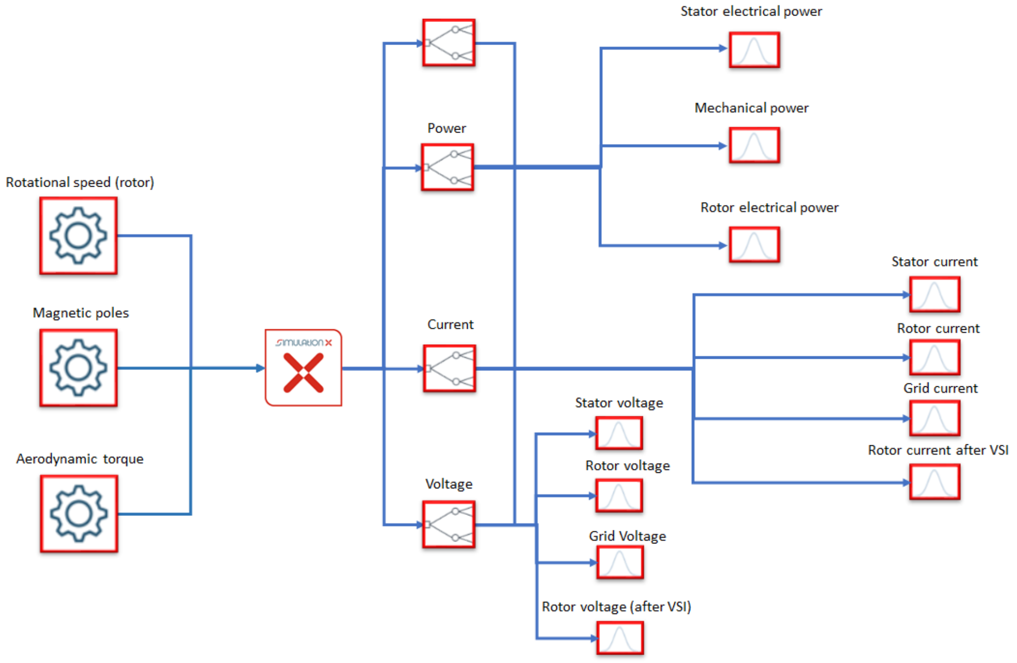
- Model-exchange: the model is running on the solvers from the execution tool
- Co-simulation: the model is shipped with the simulation libraries to be executed
  This is the mode we have implemented in Scilab (only importing FMI 1.0).

The standard specifies the interfaces of the model as and XML file (see the parameters in the next section), and the model executable as a binary file (DLL for instance):



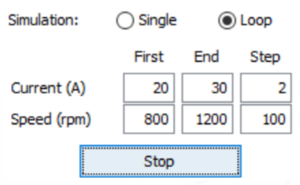
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<fmiModelDescription fmiVersion="1.0" modelName="bouncingBall" modelIdentifier="bouncingBall"
guid="{8c4e810f-3df3-4a00-8276-176fa3c9f003}" numberOfContinuousStates="2"
numberOfEventIndicators="1">
    <ModelVariables>
        <ScalarVariable name="h" valueReference="0" description="height">
            <Real start="1" fixed="true" />
        </ScalarVariable>
        <ScalarVariable name="der(h)" valueReference="1" description="velocity"> <Real/>
</ScalarVariable>
        <ScalarVariable name="v" valueReference="2" description="velocity "> <Real/>
</ScalarVariable>
        <ScalarVariable name="der(v)" valueReference="3" description="acceleration"> <Real/>
</ScalarVariable>
        <ScalarVariable name="g" valueReference="4" description="accel. gravity"
variability="parameter">
            <Real start="9.81" fixed="true" />
        </ScalarVariable>
    </ModelVariables>
</fmiModelDescription>
```

We can represent the XML Inputs & Outputs of the Functional Mock-up Unit as a graph:



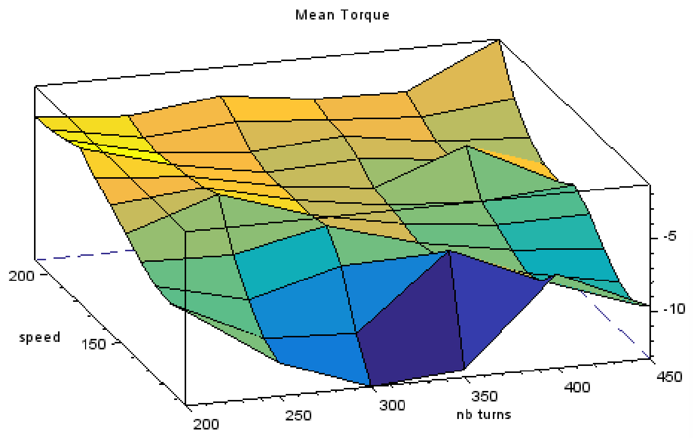
3. Optimization in Scilab

There are two different ways engineers are interested to investigate when it comes about optimization. First one is Design Space Exploration: The application was performing simulations in loop in order to build response surface as displayed below.



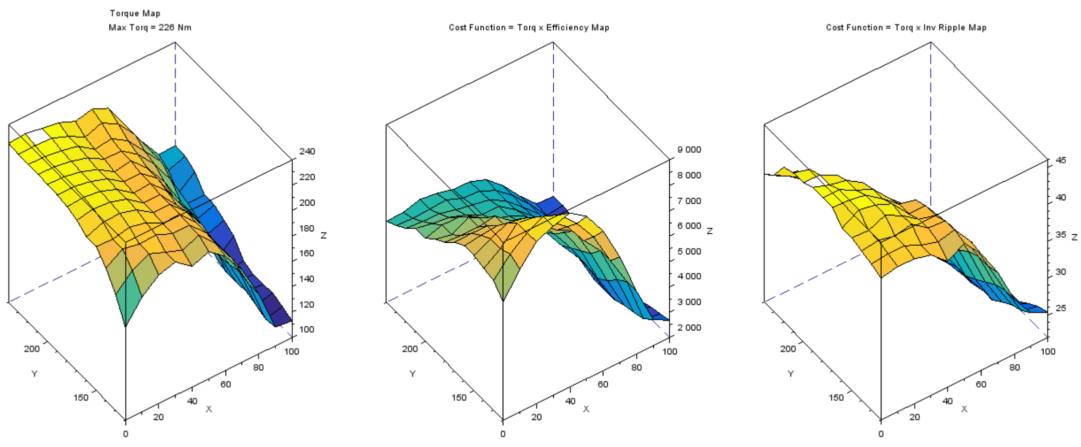
Simulation: ( ) Single [✓ Loop]

	First	End	Step
Current (A)	20	30	2
Speed (rpm)	800	1200	100



Mean Torque Response Surface built depending on speed (rpm) and number of turns

This way to proceed is pretty handy when it comes to system behavior understanding but can also be used for “manual” optimization. Although it is not suited when dealing with multiple objectives or when performing more simulations. In that case, we will have to use specific heuristics needed to first formulate a composite objective and second limit number of simulations runs needed to converge towards optimal solution.



Multi-objective optimization: Maximize torque and efficiency, minimize ripple

Dealing with models stored as Functional Mock-up Unit (FMU) allows to set-up an such an automated process as it is providing all the information needed to set-up a classical optimization problem:

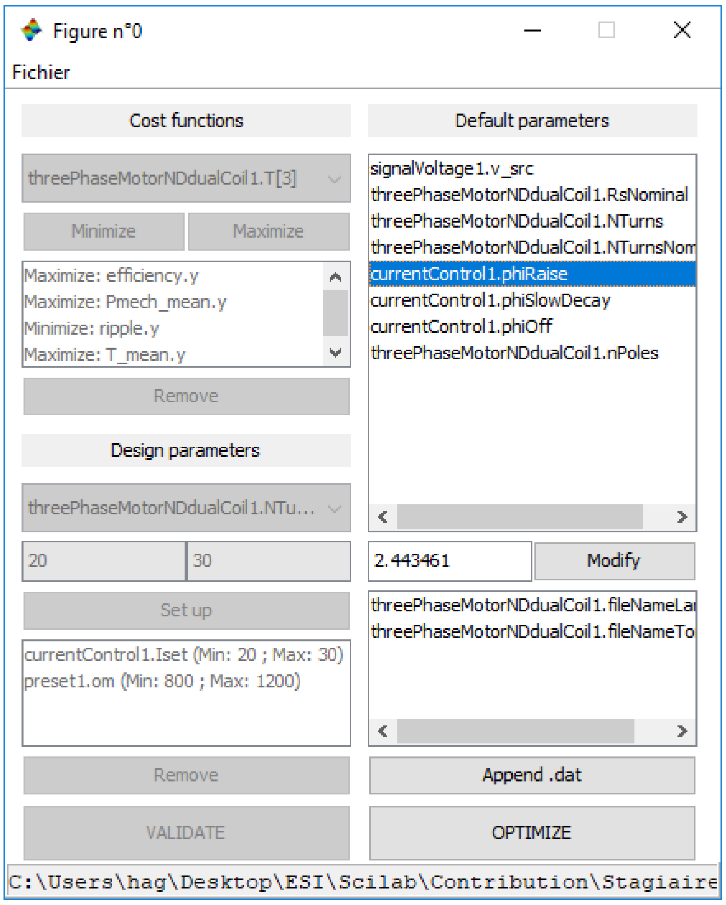
- Cost functions (causality 'output' in FMU syntax)

```
- <ScalarVariable description="Mean Torque" variability="continuous" valueReference="805306394" name="T_mean.y" causality="output">
      <Real unit="Nm" displayUnit="Nm" quantity="Torque"/>
  </ScalarVariable>
- <ScalarVariable description="Torque Ripple" variability="continuous" valueReference="805306395" name="ripple.y" causality="output">
      <Real unit="Nm" displayUnit="Nm" quantity="Torque"/>
  </ScalarVariable>
- <ScalarVariable description="Electrical Power at Supply" variability="continuous" valueReference="805306396" name="Pel_mean.y"
  causality="output">
      <Real unit="Nm/s" displayUnit="kW" quantity="Power"/>
  </ScalarVariable>
- <ScalarVariable description="Mechanical Output Power" variability="continuous" valueReference="805306397" name="Pmech_mean.y"
  causality="output">
      <Real unit="Nm/s" displayUnit="kW" quantity="Power"/>
  </ScalarVariable>
- <ScalarVariable description="Efficiency" variability="continuous" valueReference="805306398" name="efficiency.y" causality="output">
      <Real unit="-" displayUnit="%" quantity="RelMagnitude"/>
  </ScalarVariable>
```

- Design parameters (variability 'parameter' in FMU syntax)

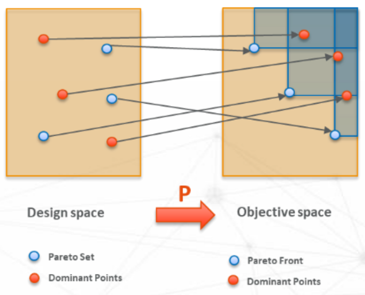
```
- <ScalarVariable description="Rotational Speed" variability="parameter" valueReference="1073741825" name="preset1.om">
      <Real unit="rad/s" displayUnit="rpm" fixed="true" start="104.71975511965977" quantity="RotVelocity"/>
  </ScalarVariable>
```

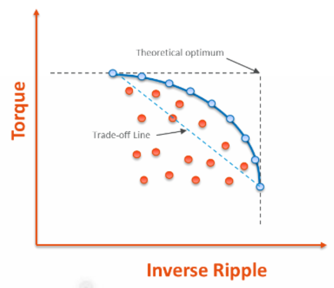
To finalize the set-up, we then need to define the goal to achieve (maximize or minimize) regarding a specific objective (Efficiency, Ripple, Mean torque, …), define the design parameters lower and upper bounds to limit the design space, and specify the remaining parameters mandatory for model simulation.



```
threePhaseMotorNDdualCoil1.T[3]

Maximize: efficiency.y
Maximize: Pmech_mean.y
Minimize: ripple.y
Maximize: T_mean.y

threePhaseMotorNDdualCoil1.NTu...
20    30

currentControl1.Iset (Min: 20 ; Max: 30)
preset1.om (Min: 800 ; Max: 1200)

signalVoltage1.v_src
threePhaseMotorNDdualCoil1.RsNominal
threePhaseMotorNDdualCoil1.NTurns
threePhaseMotorNDdualCoil1.NTurnsNom
currentControl1.phiRaise
currentControl1.phiSlowDecay
currentControl1.phiOff
threePhaseMotorNDdualCoil1.nPoles

2.443461

threePhaseMotorNDdualCoil1.fileNameLa
threePhaseMotorNDdualCoil1.fileNameTo

C:\Users\hag\Desktop\ESI\Scilab\Contribution\Stagiaire
```

The above GUI presents an optimization process set-up for FMU model. Once the case is finally set-up, a multi-objective optimization can be run (with genetic algorithms), in order to automatically build a pareto front of design solutions:

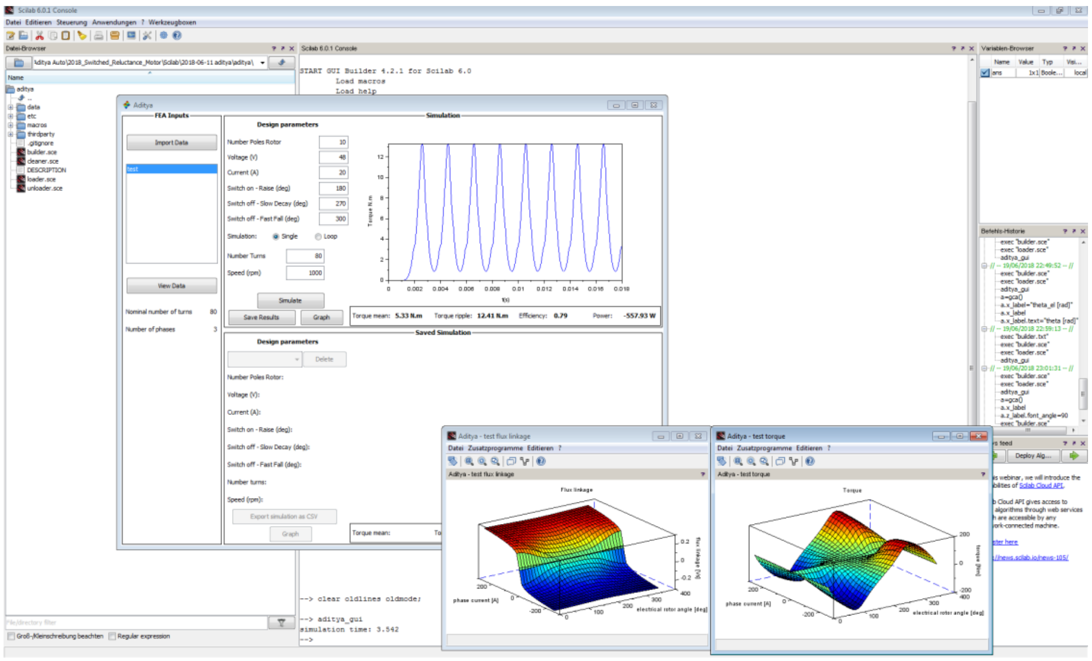




4. Application development in Scilab

4.1. Optimization application

The project presented here aimed at providing an application for the customer to optimize the parameters of the design. This application is highlighted here:



```
START GUI Builder 4.2.1 for Scilab 6.0
        Load macros
        Load help
```

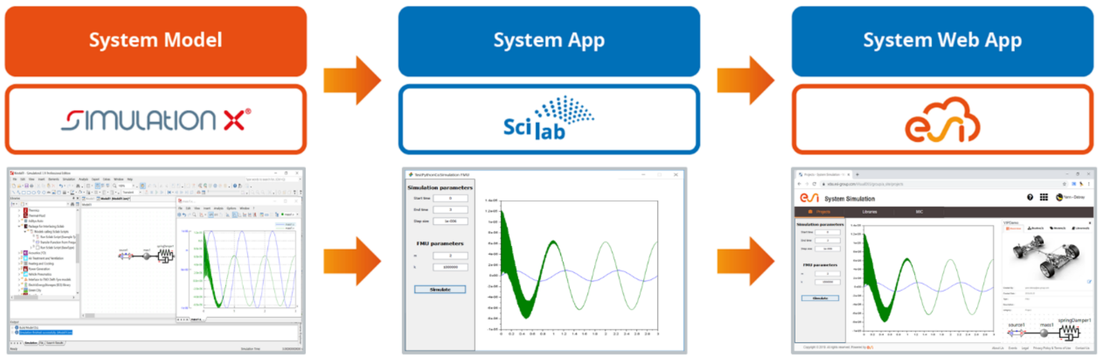
Number Poles Rotor	10
Voltage (V)	48
Current (A)	20
Switch on - Raise (deg)	180
Switch off - Slow Decay (deg)	270
Switch off - Fast Fall (deg)	300
Simulation:	[✓ Single] Loop
Number Turns	80
Speed (rpm)	1000
Nominal number of turns	80
Number of phases	3

Torque mean: **5.33 N.m** Torque ripple: **12.41 N.m** Efficiency: **0.79** Power: **-557.93 W**

```
--> clear oldlines oldmode;

--> aditya_gui
simulation time: 3.542
-->
```

4.2. Cloud deployment

The following workflow highlights the 3 main steps towards the deployment of a system application on the cloud:



Sources:

Finite Element Method Magnetics : InductionMotorExample
David Meeker dmeeker@ieee.org August 20, 2004
http://www.femm.info/wiki/InductionMotorExample

Electro-Magnetics with FEMM Finite Element Method Magnetics open-source software
https://www.scilab.org/electro-magnetics-femm

MODELISAR consortium: https://itea3.org/project/modelisar.html

Functional Mockup Interface (FMI) – General Standard for Model Exchange and Simulator Coupling – Linköping University 2011-02-10 Peter Fritzson

Xcos FMU Wrapper on Scilab Forge: http://forge.scilab.org/index.php/p/fmu-wrapper/
FMI toolbox in Scilab: https://www.scilab.org/software/atoms/fmi-model-exchange-co-simulation

Tutorial on the development of a graphical user interface in Scilab:
http://scilab.io/how-to-develop-a-graphical-user-interface-tutorial/

Development of a Switched Reluctance Motor-based Electric AC Compressor Drive
Jaehyuck Kim et al. - Journal of Magnetics, Vol. 19, No. 3, September 2014

Matlab-Simulink® Coupling to Finite Element Software for Design and Analysis of Electrical Machines – Gaizka Almandoz, Gaizka Ugalde, Javier Poza and Ana Julia Escalada http://dx.doi.org/10.5772/46476