Sophemis 4
Designer Tutorial

http://optimisation.l2ep.ec-lille.fr/download/Sophemis4_Designer_Tutorial.pdf

Table of contents

I. Introduction

In this tutorial, you will create a Sophemis model for the sizing of a surface-mounted permanent magnet synchronous motor without slot. A description of the model and some associated design issues can be found in an article in « Techniques de l'Ingénieur » (Nogarede, D3411).

The first part of this tutorial is dedicated to the understanding of the structure and content of a Sophemis model. The creation of the model for the dimensioning of this motor is done in the second part.

II. Understanding of a Sophemis model

We take the example of the model « BuildingElectricSystem » used during the user tutorial.

1. Declarative part

- Open the file « BuildingElectricSystem.m » that is in the folder « Documents\Sophemis4\models » with Matlab editor.

- A Sophemis model is a class whose name is the same as the file name without the extension (.m). The constructor of the object has the same name (red frames).

- The declarative part starts by declaring the name of the model and a comment (blue frames).

```
classdef BuildingElectricSystem < SoModel

    methods

        % declarative part

        function obj = BuildingElectricSystem

            % create object and declare name and version
            obj = obj@SoModel ('building electric system','used in tutorial');
```

- The rest of the declarative part allows to inform the interface about the inputs and outputs of the model. When creating the model, no preconceived ideas are made about the nature of the inputs (constants or variables) and outputs (objective, constraint, target or useless). This way, a model can be used with several optimization problems without modification.

- For the inputs and outputs, a name, a comment and a unit (red frames) are defined. It is also possible to define the number of values as any input or output can be a vector. If the dimension is not defined, it is assumed to be 1. This information will be visible in the interface. The comment, the unit, and the number of values appear when the mouse cursor is over the name.

- The inputs can be of the following types (blue frames):

    - **constant**: cannot change during the optimization
    - **discrete**: can only take discrete values from a set of possible values
    - **relaxable**: takes a discrete value from a set at the end of the optimization but can be continuous (relaxed) during the optimization
    - **continuous**: takes a real value in a range

- It is possible to give default values that appear when defining the optimization problem and to define some abilities such as verbose mode (green frames):

```
% declare inputs of model
obj.addInput('Psto', struct('name', 'Psto', 'comment', 'power from storage', ...
    'unit', 'W', 'type', 'continuous', 'dimension', 24));
obj.addInput('Pload', struct('name', 'Pload', 'comment', 'power consumption', ...
    'unit', 'W', 'type', 'constant', 'dimension', 24, ...
    'default', [500 500 500 500 500 500 1000 500 500 500 500 500 2000 500 ...
    500 500 500 500 1000 2000 1000 1000 500 500]'));
…

% declare outputs of model
obj.addOutput('Ppv', struct('name', 'Ppv', 'comment', 'photovoltaic production', ...
    'unit', 'W', 'dimension', 24));
obj.addOutput('Pgrid', struct('name', 'Pgrid', 'comment', 'power from the grid', ...
    'unit', 'W', 'dimension', 24));
…

% declare abilities
obj.addAbility('verbose');
```

2. Procedural part

- This part calculates the output of the model from the input values. The inputs and outputs are in structures whose field name is the name of the variable.

- The equations of the model are in the red frame.

- In the blue frame, there is a display that will be visible in the log depending on the value of "verbose" specified in the options of the algorithm. This display is optional.

```
% procedural part

function output = compute(obj, input)

    % all inputs and outputs in column

    % power from the grid (W)
    output.Ppv = input.Pirr*input.Spv*input.epv;
    output.Pgrid = input.Pload-input.Psto-output.Ppv;

    % storage state of charge (-)
    SoC = input.SoCi-3600/input.Esto*cumsum(input.Psto);
    output.SoCf = SoC(24);
    output.SoC = SoC(1:23);

    % total cost (Euro) for 20 years (no income for negative grid power)
    output.cinv = input.cpv*input.Spv+input.csto*input.Esto; % investment cost
    output.cener = 20*365*3600*input.cgrid'*(max(output.Pgrid,0)); % energy cost
    output.Egrid = 20*365*3600*sum(max(output.Pgrid,0)); % total energy from grid
    output.cost = output.cinv+output.cener;

    % verbose mode
    if isfield(input,'verbose') && input.verbose>=1
        fprintf('investment cost = %f € \r', output.cinv)
        fprintf('energy cost     = %f € \r', output.cener)
        fprintf('total cost      = %f € \r', output.cost)
        fprintf('Spv  = %f m² \r', input.Spv)
        fprintf('Esto = %f J \r', input.Esto)
        fprintf('Egrid = %f J \r', output.Egrid)
    end

end
```

3. Optional part

- This part makes a figure. It is optional. The display is accessible during the analysis of the optimization results from the "Evaluations" and "Solutions" windows with the "Draw" menu.

```
% optional part

function draw(obj, input, output)

    t = 1:24;
    subplot(3,2,1)
    bar(t, input.Pirr); title('Pirr')
    subplot(3,2,2)
    bar(t, output.Ppv); title('Ppv')
    subplot(3,2,3)
    bar(t, input.Psto); title('Psto')
    subplot(3,2,4)
    bar(t, [output.SoC;output.SoCf]); title('SoC')
    subplot(3,2,5)
    bar(t, input.Pload); title('Pload')
    subplot(3,2,6)
```

```
        bar(t, output.Pgrid); title('Pgrid')
    end
```

III. Creation of a model

1. Clarify the inputs and outputs

- The table below is extracted from the article in “Techniques de l'Ingénieur”. It shows the equations of the design model.

- We rewrite equations (1), (2), (6) and change the order of calculations:

$$\Delta p = \frac{\pi \cdot a}{p} \qquad (6) \qquad \text{then (4), (5)}$$

$$J_{cu} = \sqrt{\frac{E_{ch}}{\gamma_r \cdot b}} \qquad (2) \qquad \text{then (3)}$$

$$\lambda = \frac{2\pi}{\gamma_o}(1-\gamma_f)a^2(2\cdot a+b)B_e\sqrt{\gamma_r \cdot \gamma_p \cdot E_{ch} \cdot b} \qquad (1) \qquad \text{then (9), (10), (11)}$$

Modèle structural		Cahier des charges	
Contraintes structurales	n°	**Contraintes spécifiées**	n°
$\gamma_0 = \frac{2\pi}{\lambda}(1-v_f)a^2(2a+b)B_e\sqrt{v_r v_p E_{ch} b}$	(1)	$e_{min} - e \leqslant 0$	(7)
		($e_{min} = 10^{-3}$ m, $v_{f\,max} = 30$ %)	
		$v_f - v_{f\,max} \leqslant 0$	(8)
$E_{ch} = Kj_{cu} = v_r b j_{cu}^2$	(2)	**Impositions**	
$v_f \approx 0{,}75 p v_p \frac{e+b}{a}$	(3)	$\gamma_0 = 10$ N.m $J_r = 0{,}9$ T $v_r = 0{,}7$	
		$B_{fer} = 1{,}5$ T $E_{ch} = 10^{11}$ A^2.m^{-3} $\Delta_p = 0{,}05$ m	
$B_e = \frac{\ell_a J_r}{a \ln\left(\frac{a+b}{a-\ell_a-e}\right)}$	(4)	**Critères spécifiés**	
		$\mathcal{V}_u = 2\pi\frac{a}{\lambda}(2a+b-e-\ell_a)(2c+b+e+\ell_a)$	(9)
$c = \frac{\pi v_p B_e}{2pB_{fer}}a$	(5)	$\mathcal{V}_a = 2\pi v_p \ell_a \frac{a}{\lambda}(2a-2e-\ell_a)$	(10)
$p = \frac{\pi a}{\Delta_p}$	(6)	$P_J = 2\pi\rho_{cu}\frac{a}{\lambda}(2a+b)E_{ch}$ $\rho_{cu} = 0{,}018.10^{-6}$ Ω.m (résistivité du cuivre)	(11)

- In the Matlab editor, the equations are written:

```
Dp=pi*a/p;                                              % (6)
Be=la*Jr/(a*log(abs((a+b)/(a-la-e))));                  % (4)
c=pi*vp*Be/(2*p*Bfer)*a;                                % (5)
Jcu=sqrt(Ech/vr/b);                                     % (2)
vf=0.75*p*vp*(e+b)/a;                                   % (3)
ld=2*pi/vo*(1-vf)*a^2*(2*a+b)*Be*sqrt(vr*vp*Ech*b);     % (1)  ld=lambda
Vu=2*pi*a/ld*(2*a+b-e-la)*(2*c+b+e+la);                 % (9)
Va=2*pi*vp*la*a/ld*(2*a-2*e-la);                        % (10)
Pj=2*pi*0.018e-6*a/ld*(2*a+b)*Ech;                      % (11)
```

- The file that contains these equations is « equations.m » that is in the folder in the archive available at: http://optimisation.l2ep.ec-lille.fr/download/PMSM_Nogarede_files.zip.

- The variables on the left of the equations are outputs of the model and the variables on the right of the equations are inputs of the model. Thus, the inputs and outputs are, in order of appearance, and by specifying the equation number:

    - 9 outputs: Dp (6) ; Be (4) ; c (5) ; Jcu (2) ; vf (3) ; ld (1) ; Vu (9) ; Va (10) ; Pj (11)

    - 11 inputs: a, p (1) ; la, Jr, b, e (4) ; vp, Bfer (5) ; Ech, vr (2) ; vo (1)

2. Creation of the model class

- To create a new Sophemis template, it is easier to start from an existing template. You will take the model « BuildingElectricSystem.m ».

- Modify the names of the class and the constructor (red frames) which must be identical to the name of the Matlab file without the extension (.m). Modify the name of the model and the comment (blue frames). Save the class as « PMSM_Nogarede.m ».

```
classdef PMSM_Nogarede < SoModel

    methods

        % declarative part

        function obj = PMSM_Nogarede

            % create object and declare name and version
            obj = obj@SoModel ('PMSM by Nogarede', ...
                'Techniques de l''Ingénieur D3411');
```

- Declare the inputs and outputs of the model using the information given in the article from Techniques de l'Ingénieur (Nogarede, D3411):

Nomenclature des variables		Butées
a (m)	Rayon d'alésage	[0,005 ; 0,25]
b (m)	Épaisseur de bobinage	[0,001 ; 0,05]
B_e (T)	Amplitude du champ magnétique à vide	[0,1 ; 1,0]
B_{fer} (T)	Champ maximal admissible dans le fer	[0,5 ; 2,0]
c (m)	Épaisseur de culasse	[0,001 ; 0,05]
e (m)	Entrefer mécanique	[0,1 ; 5] 10^{-4}
E_{ch} ($A^2.m^{-3}$)	Coefficient d'échauffement	[1,100] 10^{10}
j_{cu} ($A.m^{-2}$)	Densité de courant dans le bobinage	[1,100] 10^{5}
J_r (T)	Polarisation rémanente des aimants	[0,05 ; 1,5]
ℓ_a (m)	Épaisseur d'aimant	[0,003 ; 0,05]
p	Nombre de paires de pôles	[1 ; 10]
Δ_p (m)	Pas polaire	[0,0005 ; 0,25]
γ_0 (N.m)	Moment maximal du couple électromagnétique	[0,1 ; 100]
λ	Coefficient de forme	[1,0 ; 2,5]
v_f	Coefficient de fuites interpolaires	[0,01 ; 0,5]
v_p	Coefficient d'arc polaire	[0,8 ; 1]
v_r	Coefficient de remplissage de la zone de bobinage	[0,1 ; 1,0]

- For the inputs, we declare if it is constant, discrete, relaxable or continuous. The number of pole pairs (p) is integer, so it is a discrete variable. For the other inputs, the choice of type is made by the creator of the model. If in doubt, declare continuous inputs. The declarative part becomes:

```
        % declare inputs of model
        obj.addInput('a',struct('name','a','comment',...
            'rayon d''alesage','unit','m','type','continuous'));
        obj.addInput('p',struct('name','p','comment',...
```

```
    'nombre de paires de poles','unit','-','type','discrete'));
obj.addInput('la',struct('name','la','comment',...
    'épaisseur d''aimant','unit','m','type','continuous'));
obj.addInput('Jr',struct('name','Jr','comment',...
    'polarisation rémanente des aimants','unit','T','type','constant'));
obj.addInput('b',struct('name','b','comment',...
    'épaisseur de bobinage','unit','m','type','continuous'));
obj.addInput('e',struct('name','e','comment',...
    'entrefer mécanique','unit','m','type','continuous'));
obj.addInput('vp',struct('name','vp','comment',...
    'coefficient d''arc polaire','unit','-','type','continuous'));
obj.addInput('Bfer',struct('name','Bfer','comment',...
    'champ maximal admissible dans le fer','unit','T','type','constant'));
obj.addInput('Ech',struct('name','Ech','comment',...
    'coefficient d''échauffement','unit','A2.m-3','type','constant'));
obj.addInput('vr',struct('name','vr','comment',...
   'coefficient de remplissage du bobinage','unit','-','type','constant'));
obj.addInput('vo',struct('name','vo','comment',...
    'couple électromagnétique maximal','unit','N.m','type','constant'));

% declare outputs of model
obj.addOutput('Dp',struct('name','Dp','comment',...
    'pas polaire','unit','m'));
obj.addOutput('Be',struct('name','Be','comment',...
    'amplitude du champ magnetique à vide','unit','T'));
obj.addOutput('c',struct('name','c','comment',...
    'épaisseur de la culasse','unit','m'));
obj.addOutput('Jcu',struct('name','Jcu','comment',...
    'densité de courant dans le bobinage','unit','A.m-2'));
obj.addOutput('vf',struct('name','vf','comment',...
    'coefficient de fuites interpolaires','unit','-'));
obj.addOutput('ld',struct('name','ld','comment',...
    'coefficient de forme','unit','-'));
obj.addOutput('Vu',struct('name','Vu','comment',...
    'volume utile','unit','m3'));
obj.addOutput('Va',struct('name','Va','comment',...
    'volume des aimants','unit','m3'));
obj.addOutput('Pj',struct('name','Pj','comment',...
    'pertes dans le bobinage','unit','W'));
```

- All the files containing the equations and display functions are in the compressed folder available through the link: http://optimisation.l2ep.ec-lille.fr/download/PMSM_Nogarede_files.zip.

- Create a subfolder « Documents\Sophemis4\models\PMSM_Nogarede_files » and unzip all the files in this folder.

- In the declarative part, add the line below to allow access to the files:

```
% declare file dependency
obj.addFileDependency('PMSM_Nogarede_files');
```

- Modify the procedural part as in the figure below.

```
% procedural part

function output = compute(obj, input)

    % convert input structure to variables
    fnames = fieldnames(obj.inputs);
    for i=1:length(fnames)
        eval([fnames{i} '=input.' fnames{i} ';']);
    end
```

```
        equations % compute equations

        % convert output variables to structure
        fnames = fieldnames(obj.outputs);
        for i=1:length(fnames)
            output.(fnames{i}) = eval(fnames{i});
        end

    end
```

- In the red frame are the instructions that convert an input structure into a set of variables. This is equivalent to:

```
a = input.a; b = input.b; ...
```

- In the blue frame are the instructions that convert a set of output variables into a structure. This is equivalent to:

```
output.Dp = Dp; output.Be = Be; ...
```

- Between the two frames, there is the execution of "equations.m" which contains all the equations to calculate the output variables from the input variables.

- The optional part allows you to draw a simplified three-dimensional representation of the motor:

```
% optional part

function draw(obj, input, output)

    hold on
    motordesen(input.a,input.b,output.c,output.c,input.e,input.la, ...
        input.a/output.ld,input.p,input.vp); % Courtesy of Victor Mester
    hold off

end
```

3. Validation of the model

- The first thing to do is to test the model with a set of input values for which the output values are known. We extract this information from the article in Techniques de l'Ingénieur (Nogarede, D3411):

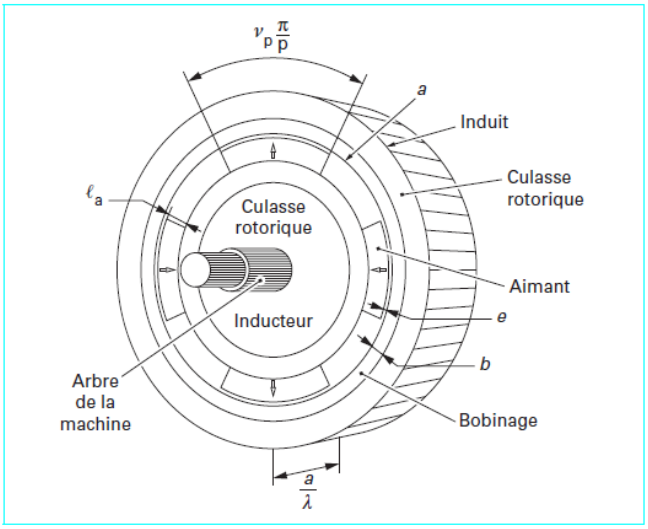


Figure 9 – Structure d'une machine tournante à aimants permanents sans encoches

● Les **spécifications imposées** à titre d'exemple consistent à dimensionner une machine capable de développer un couple γ_0 de 10 N.m, utilisant des aimants de type terres rares (J_r = 0,9 T) et des tôles supportant un champ limite de 1,5 T.

La zone de bobinage, en raison de la forme des conducteurs et des isolants mis en jeu, sera remplie à 70 % par le cuivre. L'échauffement admis est fixé à une valeur telle que le produit Kj_{cu} est de 10^{11} A^2.m^{-3}. Le pas polaire doit être de l'ordre de 50 mm.

● En raison de **contraintes** d'ordre technologique, l'entrefer mécanique doit être supérieur ou égal à 1 mm. Les fuites interpolaires seront quant à elles limitées à 30 % du flux total engendré par l'inducteur.

La transcription des spécifications du cahier des charges conduit donc, d'une part, à imposer certaines variables aux valeurs spécifiées et, d'autre part, à introduire deux contraintes de type « inégalité » (7) et (8).

● Les **objectifs de conception** sont tels que la machine recherchée doit bénéficier d'un volume des parties actives $\mathcal{V}_u$, d'un volume d'aimant $\mathcal{V}_a$ et de pertes par effet Joule P_J aussi faibles que possible.

L'expression de ces trois **critères** en fonction des variables du problème correspondent respectivement aux relations (9), (10), (11) données dans le tableau **4**.

● Les **relations de butées** définissant le domaine d'intérêt (inclus dans le domaine admissible du modèle structural) sont par ailleurs précisées sur ce même tableau.

■ Le modèle dimensionnant considéré fait donc intervenir 11 variables impliquées au sein de 8 contraintes à satisfaire et 3 critères à minimiser. La résolution du problème vis-à-vis de chacun des critères considérés individuellement tout d'abord, puis simultanément grâce à la méthode de Marglin [8] (volume des parties actives considéré comme critère principal), a conduit à 4 **dimensionnements** différents détaillés dans le tableau **5**. L'allure des machines correspondantes est représentée sur la figure **10**.

- The left column, for which the useful volume is minimized, is used. Take care that the row v_r is in fact v_p

Variable	**Critère**			
	$\mathcal{V}_u$	$\mathcal{V}_a$	P_J	$\mathcal{V}_u$, $\mathcal{V}_a$, P_J
a (m)	0,0796	0,0796	0,0637	0,0637
b (m)	0,0055	0,0054	0,0031	0,0040
B_e (T)	0,287	0,289	0,633	0,521
c (m)	0,0038	0,0039	0,0105	0,0069
e (m)	0,001	0,001	0,001	0,001
J_{cu} (10^6A.m^{-2})	5,096	5,143	6,788	5,976
p	5	5	4	4
ℓ_a (m)	0,0030	0,0030	[0,0169 ; 0,0190]	0,0076
λ	2,4935	2,4996	2,4970	2,1160
v_f	0,245	0,241	0,193	0,188
v_r	0,8	0,8	1,0	0,8
Critère optimal				
$\mathcal{V}_u$ (10^{-4}m^3)	**[5,511 ; 5,512]**	[5,527 ; 5,528]	[7,565 ; 7,796]	[6,121 ; 6,122]
$\mathcal{V}_a$ (10^{-4}m^3)	[0,742 ; 0,743]	**[0,7404 ; 0,7405]**	[2,933 ; 3,234]	[1,352 ; 1,353]
P_J (W)	[59,463 ; 59,464]	[59,282 ; 59,283]	**[37,581 ; 37,582]**	[44,664 ; 44,665]

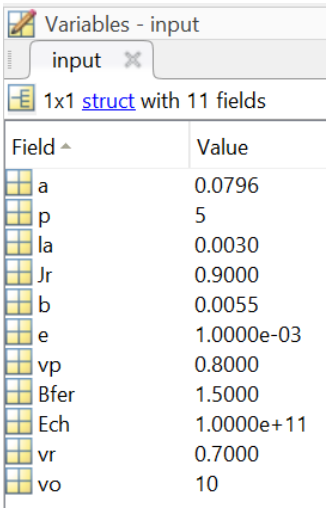
- To validate the model with Matlab, you must download the file « SoModel.p » that is available at the link: http://optimisation.l2ep.ec-lille.fr/download/core/ in the folder "core". Copy the p-file in the folder « Documents\Sophemis4\models ».

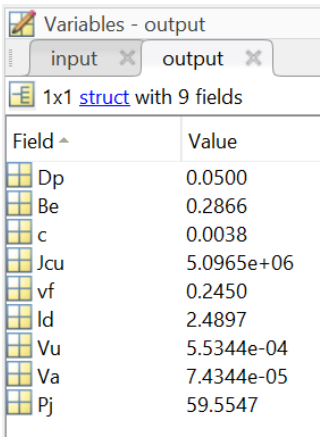
- To validate the model, enter the instructions below in Matlab Command Window or create a script in Matlab editor:

```
clc
clear variables

myModel = PMSM_Nogarede; % create model object
addpath(myModel.fileDependencies{end}); % add model subfolder
```

```
input.a = 0.0796; input.p = 5; % model input values
input.la = 0.003; input.Jr = 0.9; input.b = 0.0055;
input.e = 0.001; input.vp = 0.8; input.Bfer = 1.5;
input.Ech = 1e11; input.vr = 0.7; input.vo = 10;

output = myModel.compute(input); % test compute
```

- The workspace allows you to check the values of the outputs:

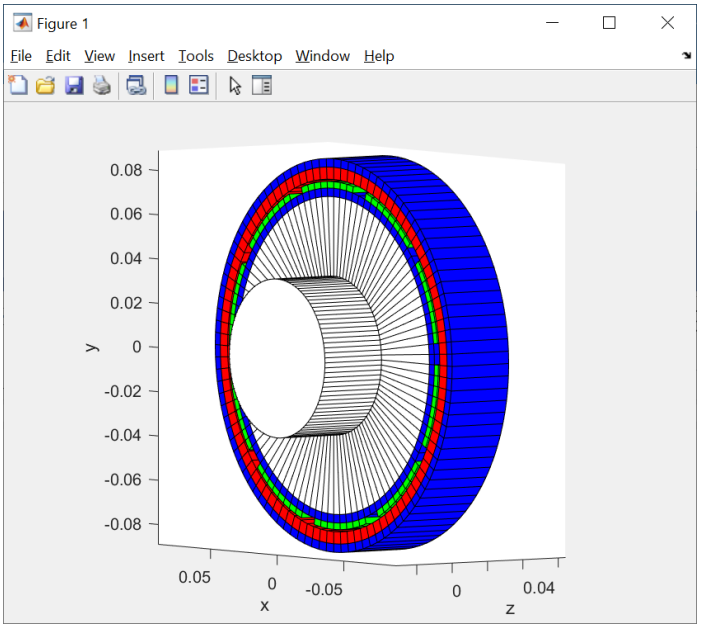




- To display the result of the draw method, type the text below:

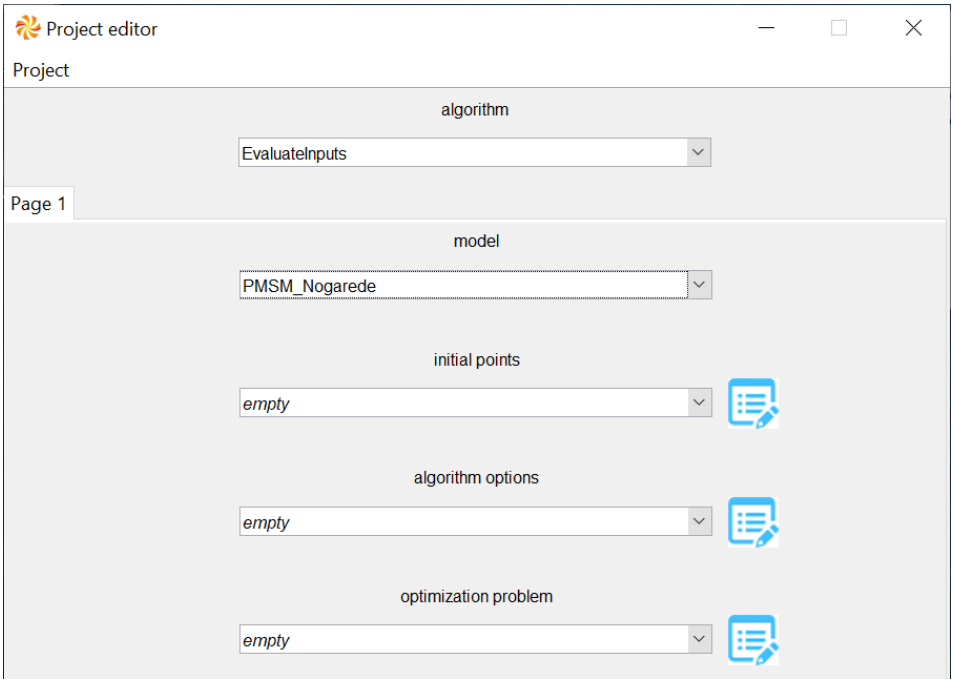
```
myModel.draw(input, output); % optional: test draw
rmpath(myModel.fileDependencies{end}); % remove model subfolder
```

- The figure below appears:

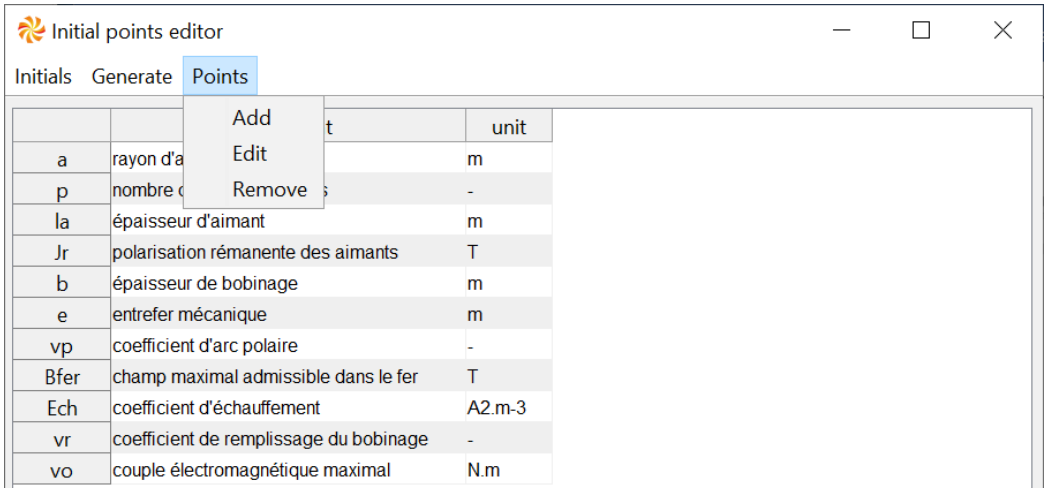


- The methods compute, draw and the declaration of the dependencies (subfolder) are validated. Delete the file "SoModel.p" in the folder « Documents\Sophemis4\models ». The next step is to validate the rest of the declarative part of the model with Sophemis.

- To validate the model with Sophemis, run Sophemis client, synchronize, and then create a new project by selecting the "EvaluateInputs" algorithm and the "PMSM_Nogarede" model:

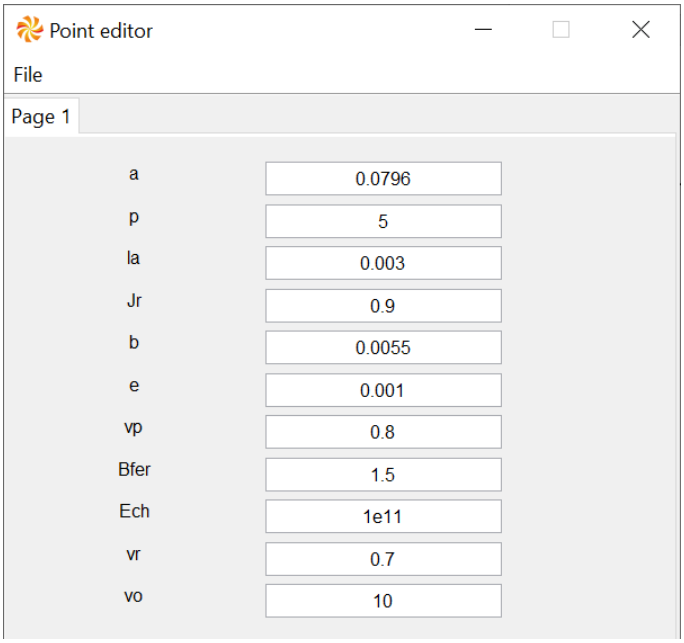


- Click on the icon  at the right of « initial points ». A window titled « Initial points editor » popups. Validate that all the input variables are presents and check their comments and units. In the menu, select « Points » then « Add »:



- If the window does not appear, an error message should appear in the console. It contains information that will allow you to locate and correct the problems. They should be located in the declarative part of the model.

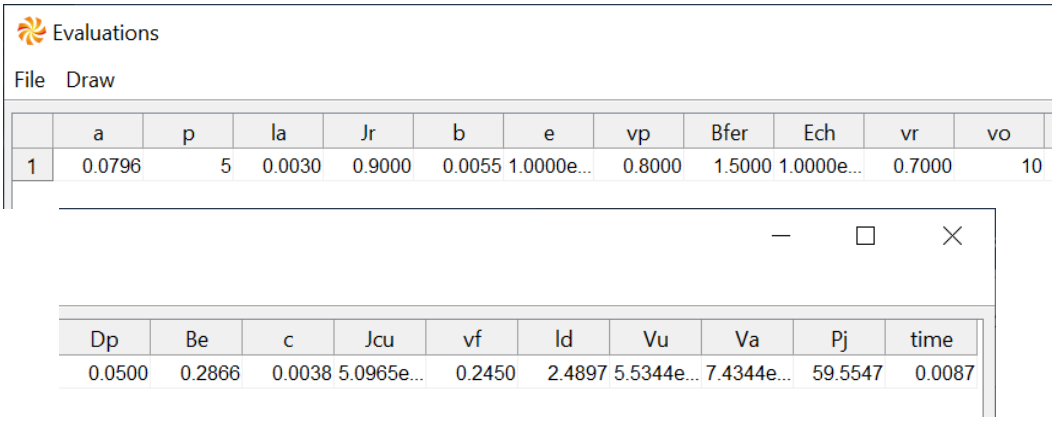
- Enter the left column values that correspond to the Vu criterion (useful volume) then save under the name « minVuValues ».



- Save the project as « MinVuValuesEvaluate ».

- A new project named « PMSM_Nogarede\ MinVuValuesEvaluate » appears in the frame « Projects » in the left side of Sophemis 4 window. Run this project. It then appears in the central frame and the right frame.

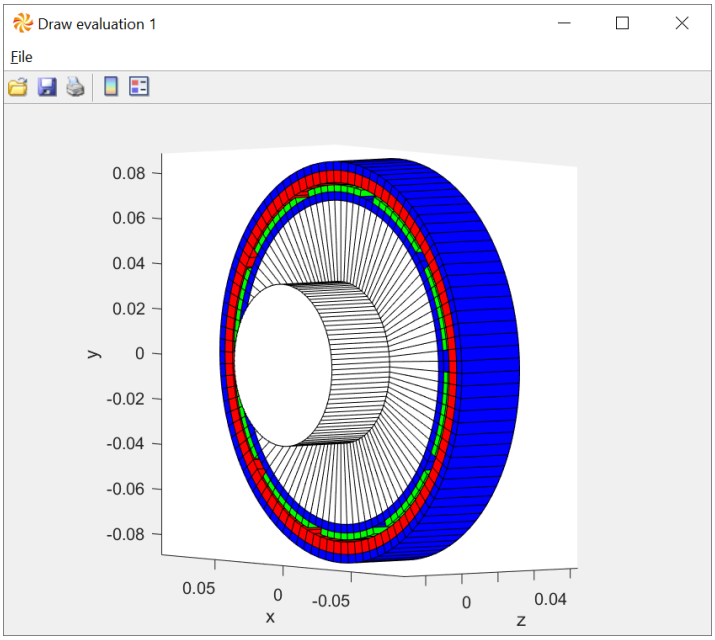
- Right-click on it and select "evaluations" to display all input and output values. Verify that the inputs are equal to the values entered and that the outputs are equal to those in the left column and the yellow highlighted text of the item.



- If "evaluations" does not appear in the pop-up menu or the "Evaluations" window is empty, click on "log" to see the error message. It contains information that will allow you to locate and correct the problems. They should be located in the procedural part of the model.

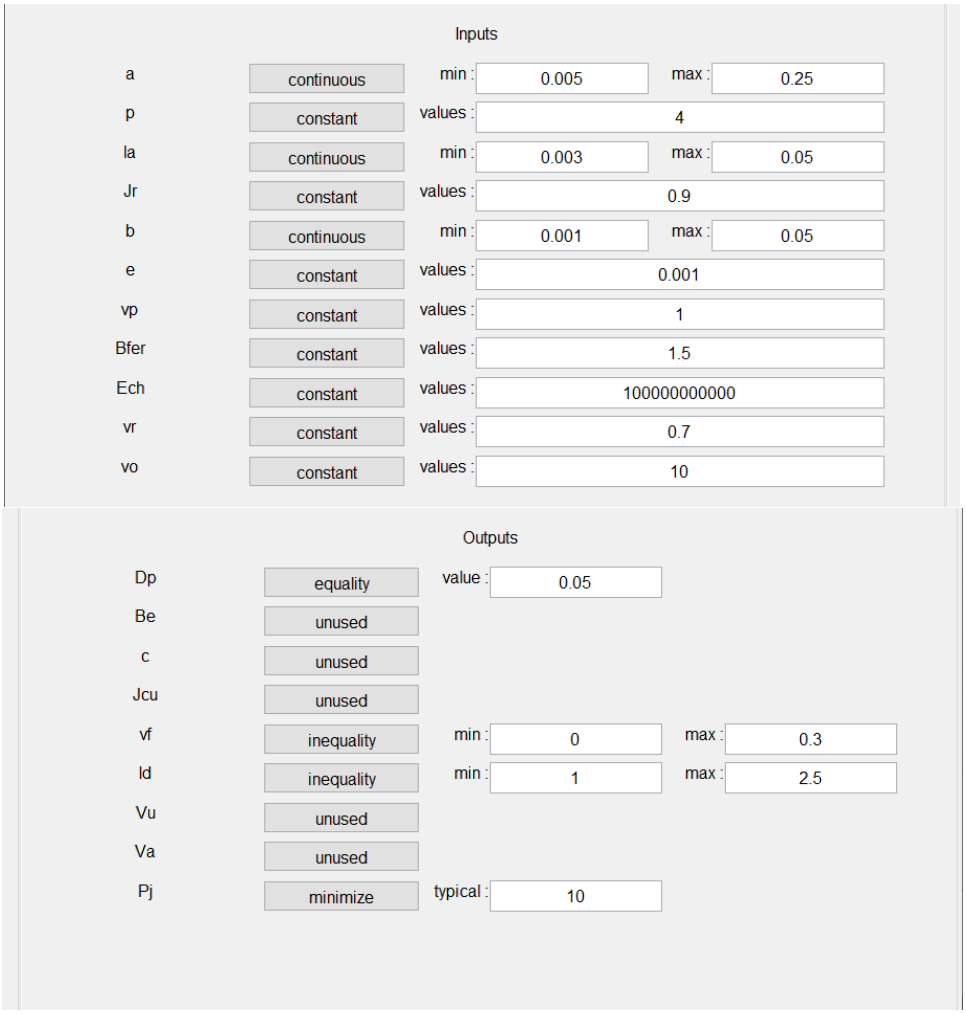
- In the "Evaluations" window, click on "Draw" in the menu. A figure appears to display the drawing of the engine corresponding to the input and output variables of the model.

- If the window does not appear, an error message should appear in the console. It contains information that will allow you to locate and correct the problems. They should be located in the optional part of the model.



4. Optimize with the model

- We now turn to the minimization of the copper loss (P_J) by varying a, la and b with constraints on Dp, vf and ld. The optimization problem is below:



- Use the “GlobalSoFmincon” algorithm with 10 workers, 100 randomly sampled initial points (trials) and a maximum of 1000 evaluations per trial.

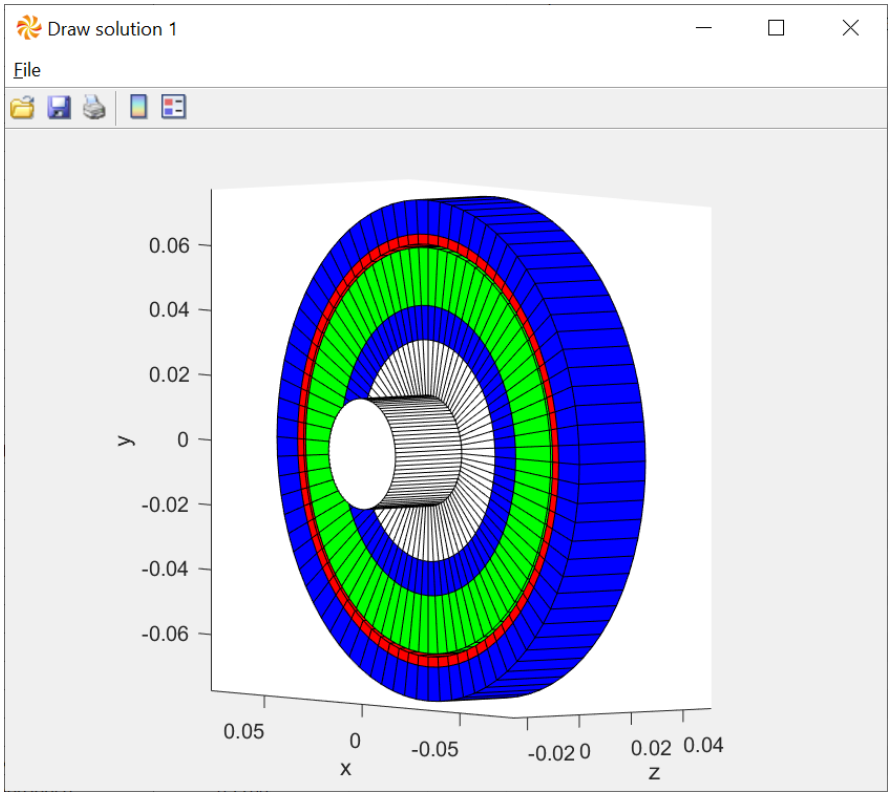
- After running the optimization, we obtain the following result which is slightly better than expected (cf. P_J column of the results table from the article). The high convergence rate highlight that the global optimum is quite easy to find.

Solutions

File Draw

	value	type	min	max	comment	unit
p	4	constant			nombre de paires de poles	-
Jr	0.9000	constant			polarisation rémanente des aimants	T
e	1.0000e-03	constant			entrefer mécanique	m
vp	1	constant			coefficient d'arc polaire	-
Bfer	1.5000	constant			champ maximal admissible dans le fer	T
Ech	1.0000e+11	constant			coefficient d'échauffement	A2.m-3
vr	0.7000	constant			coefficient de remplissage du bobinage	-
vo	10	constant			couple électromagnétique maximal	N.m
a	0.0637	continuo...	0.0050	0.2500	rayon d'alesage	m
la	0.0179	continuo...	0.0030	0.0500	épaisseur d'aimant	m
b	0.0031	continuo...	1.0000e...	0.0500	épaisseur de bobinage	m
Dp	0.0500	equality			pas polaire	m
Be	0.6328				amplitude du champ magnetique à vide	T
c	0.0105				épaisseur de la culasse	m
Jcu	6.7798e+06				densité de courant dans le bobinage	A.m-2
vf	0.1936	inequality	0	0.3000	coefficient de fuites interpolaires	-
ld	2.5000	inequality	1	2.5000	coefficient de forme	-
Vu	7.6915e-04				volume utile	m3
Va	3.0766e-04				volume des aimants	m3
Pj	37.5644	minimize			pertes dans le bobinage	W
time	0.1631					
x_1	0.2394					
x_2	0.3170					
x_3	0.0430					
f	3.7564					
constrviolation	1.1102e-15					
g_1	-0.3547					
g_2	-0.1936					
g_3	1.1102e-15					
g_4	-1.5000					
h	2.2204e-16					
exitflag	1					
convergence	0.7700					

- Draw the optimal solution. Compared to the minimization of the useful volume “MinVu”, this motor with minimal copper loss have larger magnets (green color) and yokes (blue color).



This Sophemis 4 Designer tutorial is finished. You can now create models for the devices you want to optimize. You may then proceed to the Sophemis 4 Expert tutorial where you will learn how to build a new algorithm.