Lab 8: Multi-objective optimization of a Surface Permanent Magnet electric motor

Problem Statement

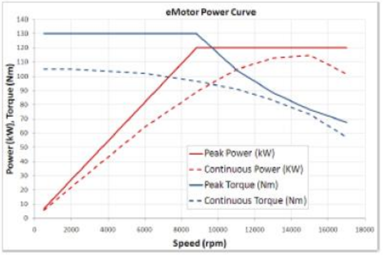
- Surface Permanent Magnets (SPM) synchronous motor for automotive application
- **Design variables**: height and width of the PMs, number of plates for each rectangular conductor
- **Objective functions**: Maximise motor efficiency and minimise the volume

Key Features

- Permanent magnet synchronous motor
- SPM type
- High power density

Performance

- Power and torque 120kW / 130Nm (transient)
  110kW / 110Nm (continuous)
- Maximum speed 17,000rpm
- Efficiency 96% (120kW, 13,000rpm)



Electrical

- Input type: high voltage sinusoidal 3-phase
- Nominal bus voltage 545V
- Maximum bus voltage 630V
- Minimum bus voltage 420V (for full performance)
- Leakage current <15mA

Mechanical

- Case diameter 185mm
- Case length 254mm
- Mass 26kg

Instrumentation & Diagnostics

- 1 Resolver for commutation

Cooling

- Max inlet temperature 55°C
- Min flow-rate 20l/min
- Pressure drop 0.6bar @ 20l/min
- Coolant type 50/50 water/glycol
- Max pressure 5bar

Environmental

- Air temperature -20°C to 90°C (operating)
  -20°C to 120°C (static)
- Max motor skin temperature 65°C
- Rated to IP67
- Maximum mounting vibration 30g
- Complies with the essential protection requirements of 89/336/EEC

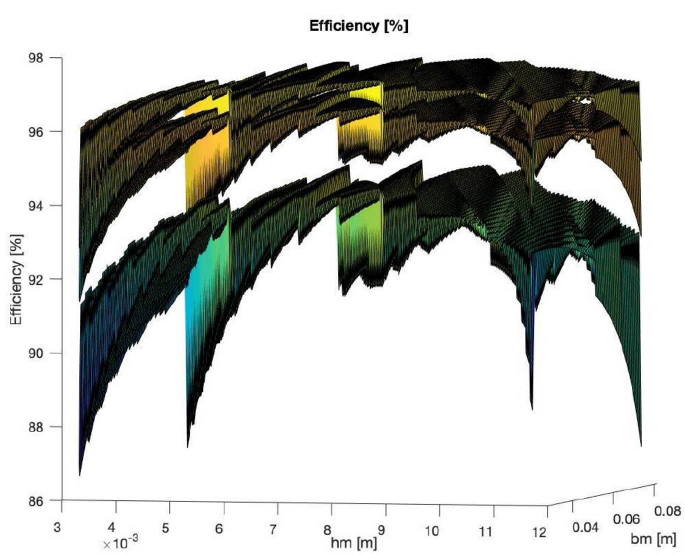
120kW / 130Nm

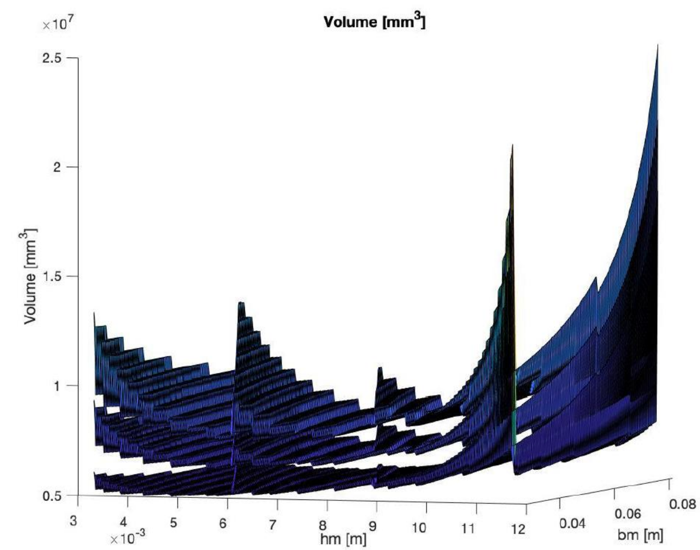
SPM motor numerical model

The matlab function “function_SPM_motor” receives as input the design variables (height and width of the PMs and number of plates) and computes the corresponding motor efficiency and volume.

[eta_NP,Volume] = function_SPM_motor(hm,bm,N_pc);

NOTE: The Matlab function handles only a single individual at time.





Design variables and objective functions

$$h_m \in [3.315, 11.715] \cdot 10^{-3} \quad [m]$$

$$b_m \in [28, 81] \cdot 10^{-3} \quad [m]$$

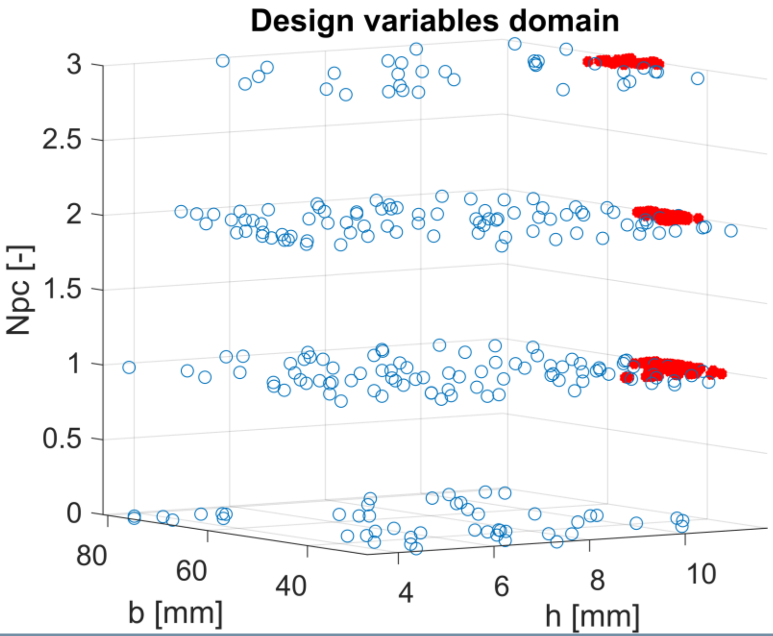
$$N_{pc} \in \{1,2,3\}$$

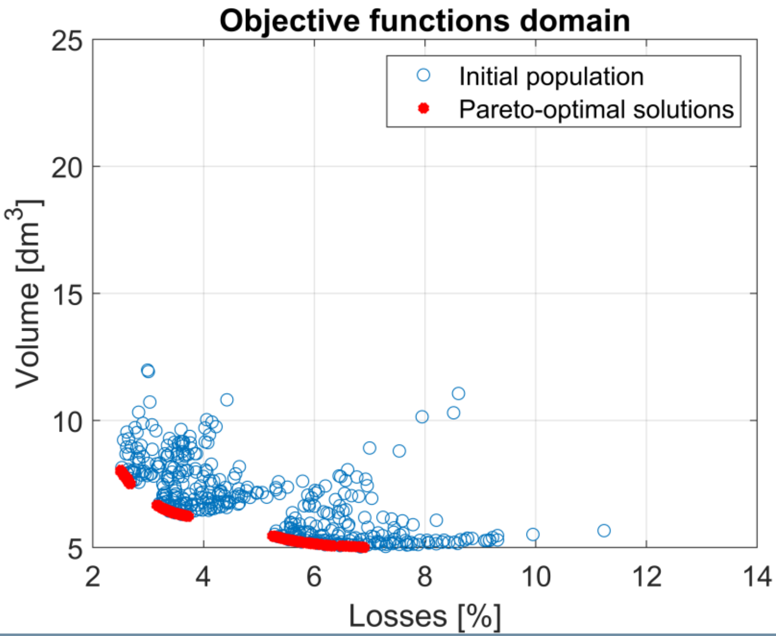
$$\eta \rightarrow Motor\ efficiency\ (to\ be\ maximised) \leftrightarrow Minimise\ Losses\ (1-\eta)$$

$$V[mm^3] \rightarrow Volume\ (to\ be\ minimised)$$

Requests

- Compute and plot the Pareto-optimal solutions of the problem in the design variables and objective functions domain
- Plot the convergence diagram of the GA algorithm (fitness VS epochs and % of individuals of rank1 in the population VS epochs)





GA algorithm: binary coding

- h_m and b_m are continuous variables, and can be treated in the same way of the previous labs
- N_{pc} can assume only discrete values (1,2,3). The simplest way is to use a 2-bit coding with the range [0, 3] and penalise unfeasible values (0).

NOTE: GA operators have been modified for this purpose. The updated versions can be found in the lab folder on the course website.

GA algorithm: fitness and constraint computation

- Fitness of individuals can be calculated according to the rank-domination criterion (the same used for the previous lab). However unfeasible solutions (i.e. individuals with N_{pc}=0) need to be penalised.
- The simplest way is to force the rank of such individuals to a high value before computing the fitness.

GA cycle example

```
% Design variables range
hm_range = [3.315e-3,11.715e-3];     % PM height [m]
bm_range = [28e-3,81e-3];            % PM width [m]
Npc_range = [0,3];                   % Number of plates of each conductor (DISCRETE
VARIABLE -> only 1,2,3 allowed)

% Genetic algorithm parameters
bit = […,…,2];            % vector of number of bits to code each DV
n = 500;                  % population size
Nmax = …;               % max generations
par = …;                % parameter to set mutation probability

% Initial population
DV = rand(n,length(bit));    % [hm,bm,Npc]
...

% GA cycle
while (percentage of rank1 individuals < 1 && i<=Nmax))
…………
end
```

To save computational time and avoid solutions clustering, the GA cycle can stop when all the individuals in the population have rank 1 (or a maximum number of iteration is reached)