Induction Motor Optimal Design by Use of Cartesian Product

prof. N. Zablodskiy [1], prof. J. Lettl [2], doc. V. Pliugin [3], ing. K. Buhr [4], stud. S. Khomitskiy [5]

[1] Donbas State Technical University/Automation of electro-technical systems, Alchevsk, Ukraine, *info@dmmi.edu.ua*
[2] Czech Technical University in Prague/Faculty of Electrical Engineering, Prague, Czech republic, *lettl@fel.cvut.cz*
[3] Donbas State Technical University/Automation of electro-technical systems, Alchevsk, Ukraine, *vlad.plyugin@gmail.com*
[4] Czech Technical University in Prague/Faculty of Electrical Engineering, Prague, Czech republic, *buhr@fel.cvut.cz*
[5] Donbas State Technical University/Automation of electro-technical systems, Alchevsk, Ukraine, *stas.blitzkrieg@mail.ru*

Abstract — **The problem of the automated calculation and optimal design of an induction motor is presented. Given and decided tasks of optimization by use of Cartesian product are introduced. The analysis of the obtained results is made.**

Keywords — Induction motor, optimization, cortege, varied variables, Cartesian product, criteria, effective variant

I. INTRODUCTION

The feature of the modern state of theory development and design practice of electric machines is passing to the automated design. Methods and facilities of automated design substantially change the character of engineering work, do them creative and effective due to considerable expansion of engineer' possibilities during realization of project researches and searching of optimum decisions. At the same time, methodology of automated design supposes high professional preparation of the modern engineer, his ability to use correctly mathematical methods and computer for making decision on the different stages of design.

Seen CAD in the context of electric machines, it is possible to distinguish following system components, used in modern electric machine design [1]:

1) automated design of electric machine;
2) searching of the optimal variant of the designed machine;
3) design of programmatic realization and searching of optimum;
4) choosing of optimum variant from a great number of effective, limitation l checking passing.

Optimal known variant of the searching methods, such as a descent method, Nadler-Mead method, method of the deformed polyhedron and others, do not allow executing of the calculations for the simultaneous change of all varied variables. As a rule, many methods assume on the contrary varying variables with subsequent adjustment of area of convergence calculations [2].

Even the simultaneous varying of all variables does not give acceptable results: a machine, which got optimum status at the found optimum value, for example, of stator package length, does not guarantee that this length will give the best result at varying of other variable, for example, diameter of stator package. There is a probability, that a machine, optimal at this diameter will be yet the best at length, different from fixed on the previous stage.

Obviously, the method of optimization, which would allow executing of the machine calculations at all possible combinations of the varied variables in the set limits and with the set step, is most acceptable.

In this paper it will be considered the method allowing to a man searching the best variant of the induction motor automated design with the set criterion of optimality, and from the great number of the expected variants to execute the selection of the best. Receipt of the possible area of project decisions, the simultaneous change of all varied variables is supposed.

II. THEORY AND PROGRAM REALIZATION

In this method the Cartesian product (CP) is a set **A × B** of all order pair of elements **(a, b)**, where **a** belongs to the **A** set, **b** to the **B** set. An order of the following pairs can be different, but the location of elements in every pair (vector) is determined by the order of the following the multiplied elements [3]. Therefore

$$A \times B \neq B \times A, \text{ if } B \neq A \quad (1)$$

A generalised form of the CP in above Eq. 1 for any number of sets $A_1, A_2, ..., A_n$, is written as

$$\prod_{i=1}^{n} A_i = A_1 \times A_2 \times A_3 \times ... \times A_n \quad (2)$$

The CP of a limited set $A_1 \times A_2 \times \ldots \times A_n$ is determined as a set of all possible sets (corteges) of the length n (made from the elements of this set), in which every element a_i belongs to the corresponding number of the set A_i. In particular, for the zero set a result is a set containing only one element - empty cortege.

Determination of CP binary operation (direct product of two sets) follows also as a special case .

For a set family $\{Xi\}_{i\in I}$ with the possibly endless indexed set I, the CP $X = \prod_{i\in I} X_i$ is possible to define as a function, comparing the element of the set X_i with every element of $i \in I$.

Let us give a simple example. Let varied in an electric motor the length of the active part L, internal D and external Da stator core diameters. A task consists of determination of such combination of the varied parameters, at which the maximum efficiency is obtained at condition of observance for the limitations laid on a project decision .

In Tab. I, the different magnitudes of the varied variables are specially given.

TABLE I
DATA SET OF THE VARIED PROJECT VARIABLES

D, mm	100	105	110	115
Da, mm	150	200	-	-
L, mm	95	99	103	-

For example, unlike to the descent method, where at first we got the decisions set at varying one variable (others are fixed), we shall set the problem of having all possible recurring combinations of the varied variables.

As a calculation result, with all data set, the real possibility to estimate adequacy of choice of optimal variant appears without some simplifying assumptions, based on all possible combinations of variables.

So the task of combination finding in some sets tends to the CP task. We will go now back to a Tab. I, where the varied variables of an electric machine are given. We will present the corteges (sets) of data for the D, Da and L sets, using examples given above.

D = {100, 105, 110, 115},
Da = {150, 200},
L = {95, 99, 103}.

We shall get 24 corteges in CP D×Da×L =
{(100, 150, 95), (100, 150, 99), (100, 150, 103), (100, 200, 95),
(100, 200, 99), (100, 200, 103),
(105, 150, 95), (105, 150, 99), (105, 150, 103), (105, 200, 95),
(105, 200, 99), (105, 200, 103),
(110, 150, 95), (110, 150, 99), (110, 150, 103), (110, 200, 95),
(110, 200, 99), (110, 200, 103),
(115, 150, 95), (115, 150, 99), (115, 150, 103), (115, 200, 95),
(115, 200, 99), (115, 200, 103)}.

Thus, the set of all order integer pairs considered by Descartes is the example of the set product on itself.

Practical application of the CP is the Optimetrics ANSOFT Maxwell unit used in optimization [4 - 6].

We shall consider program realization of the CP on the basis of corteges, presented in Tab. I. The algorithm of the CP on the recursive function call of surplus of data in corteges is realized. Below the listing of CP class on Java is given [7].

```
class Cartesian_product{
Cartesian (){};
public Vector dest;//vector with possible combinations
public int number_elem;//number of the combined parameters
public Vector get_comb(){return dest;}
public int get_number_elem(){return number_elem;}
//Search of unrepetitive combinations
//Algorithm carthesian works of great numbers
public void combinations(Vector srs, int[] size, Vector curr, int index){
   if (index == srs.size()){
      int s = curr.size();
      Integer [] d = new Integer [s];
      curr.toArray(d);
      dest.add(d);
   }
   else{
   for (int i = 0; i < size[index]; i++){
      int n = size[index];
      int [] dim = new int[n];
      dim = (int[]) srs.get(index);
      curr.add(dim[i]);
      index++;
      combinations(srs, size, curr, index);
      index--;
      curr.remove(curr.lastElement());
   }//end of for
   } //end of else

}//end of function combinations()

//Basic data and call of function to search combinations
public void init (){
      int[] dim1 = new int[]{100, 105, 110, 115};
      int[] dim2 = new int[]{150, 200};
      int[] dim3 = new int[]{95, 99, 103};
      Vector srs = new Vector();
      srs.add(dim1);
      srs.add(dim2);
      srs.add(dim3);
      number_elem = srs.size();
      int [] size = new int[number_elem];
      size[0] = dim1.length;
      size[1] = dim2.length;
      size[2] = dim3.length;
      dest = new Vector();
      Vector curr = new Vector();
      combinations(srs, size, curr, 0);
}//end of function of init()
}//end of class

//Call of function CP calculation from the main class form
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){
      Cartesian_product prod = new Cartesian_product ();
      prod.init();
      Vector dest = new Vector();
      dest = (Vector) prod.get_comb().clone();
      int num = prod.get_number_elem();
```

```
    Integer [] temp = new Integer [num];
    String text = "";
    DefaultListModel list = new DefaultListModel();
    int index = 0;
    for (int i = 0; i < dest.size(); i++){
        temp = (Integer[]) dest.get(i);
        for (int j = 0; j < num; j++){
             text = text + Integer.toString(temp[j]) + "  ";}
        list.addElement(text);
        text = "";
        index++;
    }
    jList1.setModel(list);
    jLabel122.setText("Amount of combinations :
    " + Integer.toString(index));
}
```

To reduce the calculation time, optimization for varying of only two variables, for example: internal stator core diameter and stator core length will be assumed . The maximum efficiency and minimum of starting current were chosen as criteria of optimality in this example.

The order of optimization will be following:

1) setting the range of the varied variables;
2) setting limitations;
3) choosing the criteria of optimality and set of weight coefficients for each of criteria;
4) calling the function of the CP for the searching of the varied variable possible combinations in the set range;
5) starting with the iterations number equal to a number of the CP combinations found on previous step 4);
6) calling the function of automatic electric motor calculation in the loop (number of loop steps equal to a number of the CP combinations) and function of the limitation control on each step; variant, successfully passed verification on limitations, we save in a vector; at a failure, a current selection from the found combinations is sifted from as ineffective.
7) after completion of the calculation loop we call the function of the machine best variant search among those which passed limitations and were added to the vector; we take into account the chosen criteria of optimality and their weight in the optimality calculation by Pareto method [2, 7];
8) to get the best variant data (with the optimum combination of the varied variables) we call at the end the function of electric motor automatic calculation .

Optimization results, based on the CP algorithm, are presented in Fig. 1 as a diagram of the average optimality criterion by Pareto.

In shown results the calculation of the CP in the example 15 sets of lengths and diameters were found, and only 10 of them passed limitations. A base variant in the diagram (Fig. 1) is presented in the first column (num. 1 on the axis of variant numbers). From the diagram it is obvious that the third set of combinations is the best, worst is the last, eleven set.

In spite of the fact that in the optimum variant (third) it has a little bit low efficiency factor in comparison with the base variant (first), it is better due to the index of starting current. By the calculation of the average criterion by Pareto this third variant will be leading.

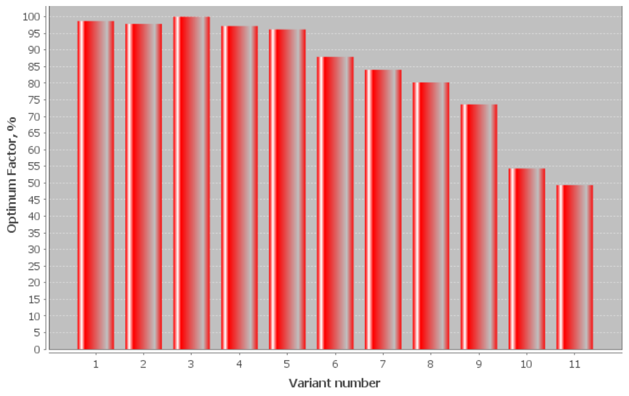

Fig. 1 - Results of multi-criterion optimization by use of the CP algorithm

The algorithm of an effective variants selection is based on the rule of Pareto preferences [2]. According to this rule, from the set of acceptable variants a variant Ko is selected. In further from Ko = 1 and for all j-criteria check up conditions:

$$F_{kj} < F_{koj}, \qquad (3)$$

where k – an index of a calculated variant; j – an index of the checked optimum criteria (at least 1 criteria must be exist).

Under the words “calculated variant” F_{kj} we are mean an array of variables selected for check up according to optimum criteria. For example, if we want to obtain the best variant with minimum starting current I and maximum efficiency η as optimum criteria, an array of 4 found variants will be looking as is shown in Tab. 2.

TABLE II
AN EXAMPLE OF VARIANTS ARRAY FOR SELECTION

Criteria (j-index)	Obtained criteria value in calculated variant (k-index)			
	1	**2**	**3**	**4**
I, A	10	12	11	9
η	0.88	0.91	0.89	0.82

Variants that not satisfied to this condition, are cast aside as wittingly «bad», because inferior to other on all criteria. From other variants a new variant is selected and got an index Ko. Then the condition (3) is checked again. A process recurs until there will be not any variant which have not got the index Ko. Remaining variants will make the set of effective variants.

The receiving of effective variants set allows considerably narrowing a searching area, but the problem of an optimal variant choice remains. If the amount of effective variants is big, the criteria wrapping is made. We will consider one of wrap methods [2].

Let F^*_j for the best value of j-criteria among effective variants and F_{kj} – the value of j-criteria in a k-variant.

Then a value of W_{kj}

$$W_{kj} = \frac{F_{kj} - F^*_j}{F^*_j} \quad (4)$$

determines how a current variant F_{kj} worse than the best F^*_j by a j-index.

Lets define as W^*_j the worst value of a variant W_{kj} and execute a normalization:

$$\hat{W}_{kj} = W_{kj} / W^*_j \,. \quad (5)$$

If to input the weight factors ξ_j for each criterion, then it is possible to form the generalized optimum criteria

$$F_\xi = \frac{1}{3}\sum_{j=1}^{3} \xi_j \cdot \hat{W}_{kj} \to \min\,. \quad (6)$$

A variant having a minimal F_ξ value (6) is near to be the best and, consequently, is an optimal with current optimal criteria weight factors ξ_j. Changing the elements of vector ξ in accordance with one or another preferences, it is possible to get the different best variants.

The Java code of function that realized represented Pareto optimum selection method is shown below.

```
public static int Pareto(Vector eff, int[] krit){
   int Nopt = 0;//optimal variant index
   opt_pareto = new Vector();//vector of optimal solution
   int Ni = eff.size();//amount of effective variants
   int Nj = krit.length;//amount of weight factors

   double[][] W = new double[Ni][Nj];//comparison values
   double[][] W1 = new double[Ni][Nj];//normalization W
   double[] Wmax = new double[Nj];//worse values W
   double[] Fmax = new double[Nj];//best values
   double[] Fw = new double[Ni];//generalized criteria
   double[] Fwp = new double[Ni];//generalized criteria in %

   for (int i = 0; i < Nj;i++){
      Fmax[i] = Double.POSITIVE_INFINITY;
   }
   //Finding of the best value for each criteria
   for (int i = 0; i < Ni; ++i){
      double[] temp = new double[Nj];
      temp = (double[])eff.get(i);
      for (int j = 0; j < Nj; ++j){
         if (temp[j] < Fmax[j]){
            Fmax[j] = temp[j];
         }
      }
   }

   //Obtaining comparison results
   for (int i = 0; i < Ni; ++i){
      double[] temp = new double[Nj];
      temp = (double[])eff.get(i);
      for (int j = 0; j < Nj; ++j){
         W[i][j] = (temp[j] - Fmax[j])/Fmax[j];
      }
   }

    //Finding of the worse value in array W
   //Worse value has a maximal divergence with the best
   for (int i = 0; i < Nj; i++){
      Wmax[i] = W[0][i];
   }
   for (int i = 1; i < Ni; ++i){
      for (int j = 0; j < Nj; ++j){
         if (W[i][j] > W[i-1][j]){Wmax[j] = W[i][j];}
      }
   }
   //Normalization
   //The worse variant has a maximal value of W1 and equal to 1.0
   for (int i = 0; i < Ni; ++i){
      for (int j = 0; j < Nj; ++j){
         W1[i][j] = W[i][j]/Wmax[j];
      }
   }
   //Calculation of the generalized optimal criteria
   //The best variant will have minimal Fw value
   //Current value is improved (decreased) by the weight factor
   //Weight factor range: from 1 (without correction) to 100 (maximal)
   for (int i = 0; i < Ni; ++i){
      for (int j = 0; j < Nj; j++){
         Fw[i] += W1[i][j]/((double)krit[j]);
      }
      Fw[i]/=Nj;
   }
   //Choosing the best variant
   double Fpmin = Double.POSITIVE_INFINITY;
   double Fpmax = Double.NEGATIVE_INFINITY;
   for (int i = 1; i < Ni; ++i){
      if (Fw[i] < Fpmin){
         Fpmin = Fw[i];
         Nopt = i;
      }
      if (Fw[i] > Fpmax){Fpmax = Fw[i];}
   }
   //Conversion of Fw value to %
   for (int i = 0; i < Ni; ++i){
      Fwp[i] = Fpmin*100/Fw[i];
    }
   //Export data for diagram drawing
   opt_pareto.add(Fwp);//Optimal variant value in %
   opt_pareto.add(Nopt);//Position of optimal variant in array

   return Nopt; //optimal variant position in input array
}
```

After selection of the best variant is finished, we are making last motor calculation with got optimal parameters D, Da and L that are associated with the optimal variant position in input data set.

III. CONCLUSIONS

Let us analyze the obtained results and make the conclusions.

1) In the CP algorithm with the varied variables range ± 1 % from a base size (15 combinations), the calculation time was 14 sec. At the range ± 10 % (1200 combinations) the calculation time grew to 12 min, that is fully acceptable. At the range of varying ± 20 % (3976 combinations) the calculation time already approached to 48 min. For the number of the varied variables changed up to 4 and to accept the range of their varying ± 20 % relatively from a base value, then we get about 1,5 million combinations and 8 hours of calculation. Thus it is needed up to 1,5 GB of computer RAM.
2) A further increase of varied variables number does not make sense, as a memory consumption sharply increases and resources of the personal computer are not enough for treatment of enormous number of variables. The preliminary estimation of the calculation time is a few days of optimization on a modern computer.
3) Thus, the CP algorithm is expedient at a small number of the varied parameters (2 - 3) and with the range of their rejection relatively to base value ± (10-20) %.
4) On the other hand, the CP algorithm in comparison, for example, to a genetic algorithm [8], allows executing of multi-criterion optimization, that is his undoubted advantage . In addition, the CP always gives the unique, i.e. only the best variant among existing ones.

The designer decides what algorithm to choose. If importance of an optimal result reception outweighs expenses on its obtaining , then one often ignores the calculation time, and it is possible to apply a difficult optimization algorithm.

When it is needed to produce approximate calculations in the maximum compressed terms, and quality of the obtained results it is in a permissible error range, then it is possible to use fast-acting, but less precision algorithms.

REFERENCES

[1] I.P. Norenkov. Automated design. M.: Informatics in an educational university, 2000. - 188 p.

[2] G.V. Reklaitis, A. Ravindran, K.M. Ragsdell. Engineering optimization. Methods and applications, part 1. Aerospace and mechanical university of Arizona.- JW&So, 1983. – 351p.

[3] N.K. Vereshchagin, A. Shen. Lectures on mathematical logic and theory of algorithms. Beginning of sets theory. MCNMO, 2008. – 198p.

[4] Three-phase induction machine. Ansoft Maxwell Field Simulator V12 – Training Manual, 2009. – 59 p.

[5] Ansoft Maxwell 2D - Electromagnetic and Electromechanical Analysis: user's guide. Ansoft corporation, 2009. – 334 p.

[6] Ansoft Maxwell 3D - Electromagnetic and Electromechanical Analysis: user's guide. Ansoft corporation, 2009. – 871 p.

[7] N. Zablodskiy, V. Pliugin, K. Buhr. CAD of electromechanic devices: educational tutorial, part 2, 2013. - 330 p. (will be printed).

[8] N. Zablodskiy, V. Pliugin, K. Buhr, J. Bauer. Induction motor design with the using of genetic optimization algorithms (will be printed).

REFERENCES ON RUSSIAN:

[1] И.П. Норенков. Автоматизированное проектирование. М.: Информатика в учебном университете, 2000. – 188 с.

[3] Верещагин Н.К., Шень А. Лекции по математической логике и теории алгоритмов. Начала теории множеств. МЦНМО, 2008. – 198с.