Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE
Informatyka w Technologii Materiałów

Vol. 16, 2016, No. 1

REAL-TIME EVOLUTIONARY OPTIMIZATION OF METALLURGICAL PROCESSES USING ARM MICROCONTROLLER

ADAM MROZEK[1]*, WACŁAW KUŚ[2], ŁUKASZ SZTANGRET[1]

[1]AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Kraków, Poland
[2]Silesian University of Technology, ul. Konarskiego 18a, 44-100 Gliwice, Poland
**Corresponding author: amrozek@agh.edu.pl*

Abstract

The real time (RT) computations with the use of microcontrollers have been present in everyday life for years. They are very useful in e.g. online control of processes due to the ability to determine the proper control in case of any environment changes. The algorithms employed in RT computation must be as simple as possible to meet the imposed time limits. On the other hand, the continuous increase in computational power of modern microcontrollers and embedded platforms causes that more complex algorithms can be performed in the real time. However, during implementation of any algorithm the specific structure and requirements of the microcontroller must be taken into consideration. Another way of fulfilling the time limits of the RT computations is application of metamodel instead of model of controlling process. Within this paper the possibility of application of evolutionary algorithm (EA) to solve three chosen optimization problems in real time using microcontroller of ARM architecture was considered. Analyzed optimization problems were as follows: aluminum alloy anti-collision side beam hot stamping process, laminar cooling of dual phase (DP) steel sheets and minimization of the potential energy of the atomic clusters. All computations were performed using two different approaches i.e. low-level and object-oriented approach. Obtained results and drawn conclusions are presented.

Key words: real time computation, microcontroller, embedded systems, optimization of metallurgical processes, evolutionary algorithm, metamodel

1. INTRODUCTION

Real time (RT) computations and processing of data using microcontrollers is nowadays widely used in many areas of applications. The real time means that the computations must be performed within a previously specified time interval. Such algorithms enable immediate reactions to the any changes of the environment. Examples of applied real time cover surgery applications (simulations of cuts – Vigneron et al., 2004, plastic surgery – Lapeer et al., 2010, suturing – Berkley et al., 2004 and other procedures), thermal simulations (Isobe et al., 2013; Kim & Cho, 1997), recovering the force and location in an impact event (Shin, 2000), part inspection in manufacturing (Jaramillo et al., 2011) and controlling elastic soft robots (Duriez, 2013), application in hardware in the loop system (Mucha, 2016).

The computational power and hardware resources of microcontrollers and embedded platforms grow rapidly every year creating new possibilities for complex and time consuming algorithms. However, the performance of the typical microcontrollers is usually still lower than average personal computer. Therefore, the RT algorithms should be designed as simple as possible and highly optimized.

The optimization algorithms need to evaluate objective function hundred or even thousand times during searching of the problem solution. It is nearly impossible to optimize when the objective function value depends on direct FEM analysis. In such a case it is justified to use the metamodel of the optimized process instead of the FEM model.

A metamodel of the process is an approximation created on the basic of al lower-level model (it is a 'model of a model'). Therefore, it is created on basis of series of offline FEM simulations. There are many techniques which can be used to build a metamodel. Some guidance on the selection of the most appropriate technique can be found in (Kusiak et al., 2015). In this paper two, most commonly used of them are described. Obtained results show that the metamodel can successfully used in RT process optimization.

As an optimization method the evolutionary algorithm was used. The implementation of EA was adjusted to requirements of the ARM architecture.

All computations were performed using ST-DISCOVERY evaluation board, equipped with the STM32F429ZIT6 microcontroller. This device integrates the ARM-CortexM4F based core (ARMv7M architecture, THUMB-2 instruction set, hardware single precision floating point unit), program and data memories, and all the necessary peripherals which make an autonomic computer system (Yiu, 2014).

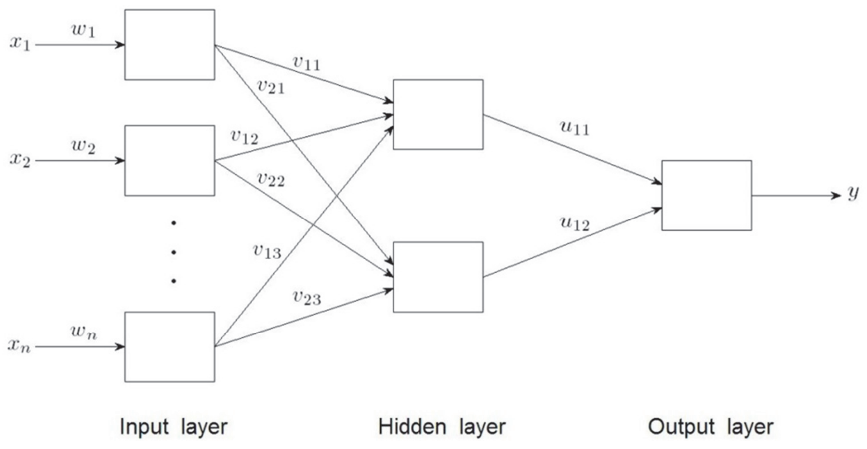


Fig. 1. *The scheme of artificial neural network.*

2. METAMODELING TECHNIQUES

2.1. Response surface methodology

The response surface methodology (RSM) is one of the most common techniques used to build metamodels. The approximation problem is formulated as a search for a function g (metamodel of the process) that gives output values close to modelled function f (model of the process). The function g is usually expressed by the linear combination of basis functions g_k multiplied by the corresponding, unknown coefficients b_k. These coefficients can be unambiguously evaluated using regression analysis if the basis functions are linearly independent and restricted to selected class. The most popular basis functions are low-order polynomials. For small curvature of the response surface, polynomial of first or second order are often used. Another class of basis functions frequently used in the RSM is class of radial basis functions (RBFs). The most popular RBFs are Gaussian, quadratic and inverse quadratic functions. More detailed description of RSM metamodelling technique can be found in (Myers & Montgomery, 1995).

2.2. Artificial neural network

An artificial neural network (ANN) is an information processing system. ANNs are composed of a defined number of interconnected artificial neurons, arranged in layers (see figure 1).

The input vector of each neuron is composed of a number of signals x_i. These signals are multiplied by the synaptic weight coefficients and their sum is an argument of the activation function. The neuron output (value of an activation function) became the input of all neurons in the following layer. The output of the neuron in the last layer is output on the network. The network topology and activation functions are selected during network creation, however the values of synaptic weights undergo variations during the supervised training process. The goal of the training is modification of weights in order to minimize the difference between the required output values (given in training dataset) and values returned by the network.

The main advantage of ANNs is their ability to learn and, as a consequence, a good capacity to approximate multivariable and nonlinear functions. Due to this and the fact that ANNs are very efficient in terms of computing time, they are very useful tool for metamodelling complex, industrial processes.

3. OPTIMIZATION METHOD

The evolutionary algorithm with floating point genes (Kuś & Burczyński, 2008; Kuś et al., 2011) were chosen as an optimization algorithm for microcontroller. The evolutionary algorithm operates on population of individuals containing chromosomes (one chromosome per individual in most cases). The chromosome is a vector containing design variables in genes. The used version of evolutionary algorithm codes each design variable using floating point gene. The presented algorithm is composed of few stages. The first stage is random generation of starting population of individuals (chromosomes). Next, the fitness function for each chromosome is evaluated. The fitness function in most problems is equal to objective function, but can be extended to incorporate some additional constraints. According to the Darwin theory of evolution of species, the individual with the highest fitness to the environment has the biggest chance to survive and proliferate. The similar approach is used in evolutionary algorithms during selection process. The algorithm presented in this paper uses tournament selection. The idea of tournament selection is grouping individuals and emerging the winner in each group on the basis of the highest fitness function value. The winner survives and undergoes modifications with the use of evolutionary operators. The typical evolutionary operators are crossovers and mutations. The simple crossover was used during computations. On the basis of randomly chosen two individuals, new individuals were created by exchanging parts of the parents' genetic material. The Gaussian version of mutation operator was applied, thus the genes were modified by adding random number with normal distribution. The number of crossovers and mutations during one step of evolutionary algorithm is defined by probability of operators.

The evolutionary algorithm for microcontroller was implemented using two approaches. The Keil uVision Integrated Development Environment was used in the first case. Since evolutionary algorithm is driven by the Random Number Generator (RNG), the 32-bit hardware RNG, built-in ST32F429ZIT6 microcontroller was used. Such generator uses analog electrical noise as an entropy source – the source of true random bits, which seeds Linear Feedback Shift Registers – the classical construction of hardware pseudo-random number generator. The routine, which services and scales the output of RNG to desired format and range, was written in assembly language (see Hohl & Hinds, 2015 for more details). Also all the configuration functions of the system (i.e. clocks, buses, CPU core and peripherals) were written by the Authors. The floating point computations were performed using hardware single-precision Floating Point Unit (IEEE 754 compliant) built in microcontroller. The rest of the algorithm was written in C language. The speed-criteria optimization was chosen during compilation. This approach allows to control all the resources of the microprocessor system and results in fast, highly-optimized code. Such approach is time-consuming, requires hardware-level knowledge and resulting code is hard to read by others. Moreover, the modifications cannot be usually applied easily.

In the second case, the opposite approach was taken into consideration. The evolutionary algorithm was implemented also using .NET Micro Framework (MF) (Kuehner, 2008). The .NET MF allows creation of software using C# language. The development environment like Microsoft Visual Studio can be used for program compiling, debugging and testing both on emulators and external hardware. The platform allows to create programs in fast way with easy porting of application on different hardware platforms. The drawback of such approach are additional layers in software between user program and hardware which leads to longer execution times of C# programs than programs developed using dedicated C compilers or assembler.

4. ANALYZED OPTIMIZATION PROBLEMS

4.1. Aluminum alloy anti-collision side beam hot stamping process

The first optimization problem solved using evolutionary algorithm was the problem of controlling the aluminum alloy anti-collision side bean hot stamping process. Optimization was performed on the ST-DISCOVERY evaluation.

The evolutionary algorithm was implemented using C# .NET MF and C language. The comparison of optimization time for chosen problem is given. The metamodel for aluminum alloy anti-collision side beam hot stamping process created by Zhou et al. (2013) was built using RSM technique and it was used in this example. The optimization based on the metamodel can be performed in real time. The created metamodel predicts three criteria: the minimum value of thickness, the rupture distance and maximum distance of the blank to die. The multi-criteria optimization problem was transform into one-criterion problem using weighted sum method with arbitrary chosen weights. The population consisted of 30 chromosomes with two genes were used during optimization. The simple crossover with probability 50% and Gaussian mutation also with probability 50% were used. The tournament selection with tournament size equal to 3 was used. The parameters of evolutionary algorithm were chosen on the basis of previous experiments for typical optimization problems in mechanics (Kuś et al., 2011).

The optimization was performed using two implementations of algorithm (one written in C and other one in C# within .NET MF framework) described in chapter 3. In both cases the CPU core clock frequency was equal to 180MHz, which is the maximal frequency for ST32F429ZIT6 microcontroller. No overdrive techniques were used. The code was loaded into the flash memory, and program's data (i.e. variables, heap and stack) was placed in microcontroller's internal RAM. Both, instruction and data caches as well as prefetching were enabled. The times of evolutionary optimization for 500 iterations were measured for both approaches and they are presented in table 1. The very high overhead for .NET MF approach can be observed. Taking into account only wall time of computations, one will chose the first approach. For most problems in industry, also the time to learn architecture of microcontroller, specific extension of C and assembler language is important due to high number of microcontroller platforms available on the market. The .NET MF approach allows to use the same environment for a large number of platforms, therefore the time needed for development and also rewriting existing programs is reduced.

Table 1. *Execution times for evolutionary algorithm.*

implementation	EA execution time [s]
C with assembler subroutines	0,2
.NET MF C# implementation	16,3

4.2. Laminar cooling of dual phase steel sheets

The second considered optimization problem was the problem of the optimization of cooling metal sheets made of dual phase steel has been analysed. DP steels are more and more widely used in automotive industry (Pietrzyk et al., 2009; Pietrzyk et al., 2010). Their structure is composed of islands of hard martensite (20-30%) in a matrix of soft ferrite (70-80%). Such a structure is obtained in the process of metal sheet manufacturing by using a three-step cooling cycle (figure 2):

- fast cooling to the temperature of the greatest speed of ferritic transformation,
- maintaining this temperature until the required percentage of the ferrite volume is reached,
- again, fast cooling is such a way as to ensure that the rest of the austenite is converted into martensite, and an amount of bainite in the final product is minimised.

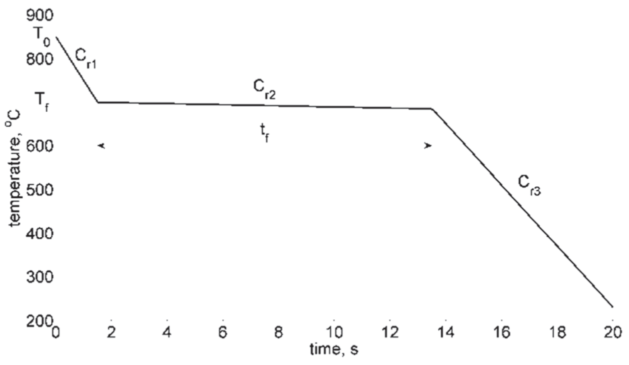


Fig. 2. *Laminar cooling scheme for DP steel.*

Such a thermal cycle is carried out either during laminar cooling after hot rolling (for thicker steel strips) or during constant annealing after cold rolling (for thinner sheets). The implementation of this technological cycle is difficult and only a few leading companies in the world have mastered the technology of manufacturing metal sheets from DP steel.

Based on the performed sensitivity analysis, two parameters have been chosen as decision variables: dwelling time at a low cooling rate within the range of ferritic transformation temperature t_f and the temperature at the beginning of the second step of the cycle, T_f . By changing these parameters, authors have tried to obtain steel with the desired percentage of volume of martensite $F_{m0} = 0.2$, bainite $F_b = 0$ and perlite $F_p = 0$. Mean squared error has been used as the objective function. During optimization the prediction of the percentage of volume of each phase was made using three artificial neural networks. To generate the training dataset the Avrami's equation based model (Pietrzyk & Kuziak, 1999) was used.

The trained ANN served as a fitness function for the evolutionary optimization. The ANN was made of 23 neurons in the following arrangement: two input neurons, twelve and eight neurons in the first and second hidden layer, respectively and finally, one output neuron.

Similarly as in the previous case, two implementations of the algorithm were developed and tested: the .NET Micro Framework and C/assembler one. Since the activation functions of the neurons involves exponential formulation, additionally tests were performed with three different implementations of the exponential function: a standard one (from the Keil uvision math.h library), assembly-coded Taylor's series approximation and highly optimized routine based-on the low-level interpretation and common memory mapping of floating point and integer variables, described by Schraudolph (1999).

In all cases, the EA was able to find desired minima (approx. T_f=693.31°C and t_f=11s, as taken from Kusiak et al., 2015). Execution times of 500 generations with 50 individuals of the EA were gathered and presented in table 2. The probabilities of the Gaussian mutation and simple crossover were set to 50%.

Table 2. *Execution times of EA + ANN*

implementation	evolutionary algorithm execution time [s]	T_f[°C]	t_f [s]
C+assembler exp from math.h	10.2s	693.74	11.07
C+assembler exp - Taylor series (look-up table)	2s	693.25	10.86
C + assembler approx. exp	1.1s	694.7	11.07
.NET MF C# implementation	4.7min	718	12.6

4.3. Minimization of the potential energy of the atomic clusters

As the third optimization problem EA was applied to the optimization of the shape of the small isolated molecular structures, called atomic clusters. Such structures contain up to several thousands of atoms and have some unique mechanical, optical and electronic properties, due to their ideal, the most stable, spatial arrangement of atoms. Such ideal configuration of atoms corresponds to the global minimum of the potential energy surface (PES) of the atomic cluster. However, searching for the global minimum on the PES is a NP-hard problem, because the number of local minima increases almost exponentially with increasing the size of cluster.

In the former author's implementation of EA (Mrozek et al., 2010), the chromosomes contained the real-valued Cartesian coordinates of each atom in the considered cluster. The total number of design variables was equal 3*N*, where 3 is the dimension of the problem and *N* denotes the number of atoms in the optimized cluster. The initial coordinates were generated randomly in the range:

$$\left[0,\ldots,r_0\sqrt[3]{N}\right], \tag{1}$$

where: r_0 is the equilibrium distance between two atoms. In such approach the space, occupied by the cluster, scales with increasing number of atoms. The fitness function was directly expressed as the total potential energy of the considered atomic cluster i.e. sum of all potential energy's contributions from interactions between two atoms:

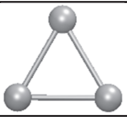
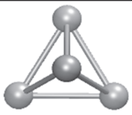
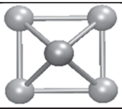
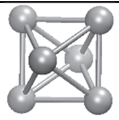
$$V_{tot} = \sum_{i,j>i} D_e \left[e^{-2\alpha\left(r_{ij}-r_0\right)} - 2e^{-\alpha\left(r_{ij}-r_0\right)} \right] \tag{2}$$

The exponential function in eq. (2) denotes the Morse potential, which determines the behavior of the atoms. The following parameters: dissociation energy D_e=0.2703eV, equilibrium distance r_0=3.253Å, and scaling parameter α=1.1646 Å^{-1}, fitted to the properties of the bulk aluminium, were taken from the work (Girifalco & Weizer, 1959).

The size of the population was ten times greater than number of design variables. All the code was written in the C language with assembly subroutines. The rest of the evolutionary algorithm's parameters remain unchanged.

Obtained results, along with times of the computations and visualizations of the molecular structures are gathered in table 3. In the all cases (N=3,…,6), implemented evolutionary algorithm was able to find global minima. Although considered atomic structures are simple and the most stable configurations are easy to predicate (Mrozek et al., 2010), presented problems (up to 18 design variables) show abilities of the algorithm to deal with more complex fitness functions.

Table 3. *Execution times for evolutionary algorithm*

execution time [s]	potential energy [-eV]	structure
6,37	0,8126	
15,13	1,6249	
29,65	2,4999	
51,54	3,5243	

5. CONCLUSIONS

It is crucial to optimize the described algorithms considering computation time because in real time applications every millisecond matters.

The evolutionary algorithms for real time optimization of metallurgical processes were presented with two approaches: the high- and low-level one. The two types of the metamodel were applied as a fitness function: based-on the simple polynomial and ANN. The measured computation times have shown high overhead of implementation with .NET MF but the good sides of such approach were also discussed. The different approaches of computation of activation functions reveal advantages of the clever, non-classical approaches of implementation of such common routines such as exponential function.

The last example shows the abilities of the presented EA of finding the global minima of the more complex functions.

The presented examples may be quite simple, but the computations are performed on a microcontroller, a device that has been typically utilized for simpler tasks (control, communication, data acquisition and processing). However new technologies, drawing along a significant increase of computational possibilities, open a wide range of new possibilities for applications of microcontrollers. Obtained results, allow to use relatively simple and cheap STM32F429 microcontroller in the real-time optimization of technological processes.

REFERENCES

Berkley, J., Turkiyyah, G., Berg, D., Ganter, M., Weghorst, S, 2004, Real-Time Finite Element Modeling for Surgery Simulation: An Application to Virtual Suturing, *IEEE Transactions on visualization and computer graphics*, 10(3), 314-325.

Duriez, C., 2013, Control of elastic soft robots based on real-time finite element method, *Robotics and Automation (ICRA)*, 3982-3987.

Girifalco, L.A.,Weizer, V.G., 1959, Application of the Morse Potential Function to Cubic Metals, *Physical Review*, 114 (3), 687-690.

Hohl, W., Hinds, Ch., 2015, *ARM Assembly Language. Fundamentals and Techniques*, CRC Press.

Isobe, Y., Watanabe, H., Yamazaki, N., XiaoWei, L., Kobayashi, Y., Miyashita, T., Hashizume, M., Fujie, M. G., 2013, Real-Time Temperature Control System Based on the Finite Element Method for Liver Radiofrequency Ablation: Effect of the Time Interval on Control, *Engineering in Medicine and Biology Society (EMBC), 35th Annual International Conference of the IEEE*, 392-396.

Jaramillo, A.E., Boulanger, P, Prieto, F., 2011, On-line 3-D system for the inspection of deformable parts, *The International Journal of Advanced Manufacturing Technolology*, 57, 1053-1063.

Kim, S. K., Cho, D. W., 1997, Real-Time Estimation of Temperature Distribution in a Ball-Screw System, *International Journal of Machine Tools and Manufacture*, 37, 4, 451-464.

Kuehner, J., 2008, *Expert .NET Micro Framework*, Apress, Berkeley.

Kusiak, J., Sztangret, Ł., Pietrzyk, M., 2015, Effective strategies of metamodelling of industrial metallurgical processes, *Advances in Engineering Software*, 89, 90-97.

Kuś, W., Burczyński, T., 2008, Parallel bioinspired algorithms in optimization of structures, *Lecture Notes on Computational Science*, 4967, 1285-1292.

Kuś, W, Długosz, A., Burczyński, T., 2011, OPTIM - library of bioinspired optimization algorithms in engineering applications, *Computer Methods in Materials Science*, 11, 9-15.

Lapeer, R.J., Gasson, P.D., Karri, V., 2010, Simulating plastic surgery: From human skin tensile tests, through hyperelastic finite element models to real-time haptics, *Progress in Biophysics and Molecular Biology*, 103, 208-216.

Mrozek, A., Kuś, W., Burczyński, T., 2010, Searching of stable configurations of nanostructures using computational intelligence methods, *Czasopismo Techniczne. Mechanika,* 107, 4-M, 85-97.

Mucha, W., 2016, Real-time hybrid simulation using materials testing machine and FEM, *Advances in Mechanics: Theoretical, Computational and Interdisciplinary Issues, Proc. of the 3rd Polish Congress of Mechanics (PCM) and 21st International Conference on Computer Methods in Mechanics (CMM)*, Gdańsk, 419-422.

Myers, R. H., Montgomery, D. C., 1995, *Response Surface Methodology; Process and Product Optimization Using Designed Experiments*, Wiley, New York.

Pietrzyk, M., Kuziak, R., 1999, Coupling the thermo-mechnical finite-element approach with phase transformation, *Computational Materials Science*, 44, 783-791

Pietrzyk, M., Kusiak, J., Kuziak, R., Zalecki, W., 2009, Optimization of laminar cooling of hot rolled DP steels, XXVIII Verformungskundliches Kolloquium, Planneralm, 285-294.

Pietrzyk, M., Madej, Ł., Rauch, Ł., Gołąb, R., 2010, Multiscale modeling of microstructure evolution during laminar cooling of hot rolled DP steel, *Achieves of Civil and Mechnical Engineering*,10, 57-67.

Schraudolph, N. N, 1999, A fast, compact approximation of the exponential function, *Neural Computation*, 11, 853-862.

Shin, E., 2000, Real-time recovery of impact force based on finite element analysis, *Computers and Structures*, 76, 621-635.

Vigneron, L. M., Verly, J. G., Warfield, S. K., 2004, Modelling Surgical Cuts, Retractions, and Resections via Extended Finite Element Method, MICCAI, 2004, LNCS 3217, Springer-Verlag Berlin Heidelberg, 311-318.

Yiu, J., 2014, *The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors*, Third Edition, Elsevier.

Zhou, J., Wang, B., Lin, J., Fu, L., 2013, Optimization of an aluminum alloy anti-collision side beam hot stamping process using a multi-objective genetic algorithm, *Archives of Civil and Mechanical Engineering*, 13, 401-411.

EWOLUCYJNA OPTYMALIZACJA PROCESÓW METALURGICZNYCH W ZASIE RZECZYWISTYM Z WYKORZYSTANIEM MIKROKONTROLERA ARM

Streszczenie

W artykule rozważano zastosowanie mikrokontrolerów w optymalizacji parametrów procesów metalurgicznych z użyciem algorytmów ewolucyjnych. Do wyznaczenia wartości funkcji przystosowania użyto metamodeli, zbudowanych w oparciu o funkcje wielomianowe i sztuczną sieć neuronową.

Jako platformę testową wykorzystano zestaw uruchomieniowy ST-DISCOVERY, wyposażony w mikrokontroler STM32F429ZIT6 (rdzeń ARM-CortexM4F, architektura ARMv7M). Jest to 32-bitowy jednoukładowy mikrokomputer, wyposażony w sprzętową jednostkę zmiennoprzecinkową pojedynczej precyzji i wszystkie niezbędne peryferia pozwalające tworzyć samodzielny system mikroprocesorowy.

Do implementacji algorytmu ewolucyjnego zastosowano dwa podejścia. W pierwszym użyto języka C ze wstawkami asemblera, a w drugim użyto środowiska programowania opartego o Visual Studio oraz bibliotekę .NET Micro Framework. W artykule przedstawiono optymalizację z użyciem metamodelu pozwalającego dobrać parametry procesu gorącego tłoczenia belki samochodowej, chłodzenia laminarnego blach ze stali DP po walcowaniu na gorąco oraz minimalizację energii potencjalnej prostych układów kilkuatomowych. Porównano czasy pracy implementacji niskopoziomowej oraz wykorzystującej współczesny język obiektowy.

Received: May 6, 2016
Received in a revised form: July 20, 2016
Accepted: July 26, 2016