DOI: 10.5604/01.3001.0012.0701

A DISTRIBUTED METHOD FOR TRANSIENT SIMULATIONS THAT DYNAMICALLY CONSIDERS SUPPLEMENTARY RESULTS FROM AUTONOMOUS SOFTWARE AGENTS

Matthias Jüttner[1], Sebastian Grabmaier[1], Jonas Rohloff [1], Desirée Vögeli[2], Wolfgang M. Rucker[1], Peter Göhner[2], Michael Weyrich[2]

[1]University of Stuttgart, Institute for Theory of Electrical Engineering, [2]University of Stuttgart, Institute of Industrial Automation and Software Engineering

Abstract. *Based on autonomous software agents capable of calculating individual numerical field problems, a distributed method for solving transient field problems is presented. The software agents are running on distributed resources connected via a network and represent a dynamic calculation environment. Communication and data exchange between multiple agents enables their collaboration and allows decisions based on distributed overall knowledge. As unique characteristics, no central unit influences the solution process at any time. The presented simulation example and its evaluated calculation process proves the method to benefit from redundant resources.*

Keywords: automatic step size control, distributed computing, software agents, transient simulation

ROZPROSZONA METODA DO SYMULACJI STANÓW PRZEJŚCIOWYCH DYNAMICZNIE UWZGLĘDNIAJĄCA DODATKOWE WYNIKI AUTONOMICZNYCH AGENTÓW PROGRAMOWYCH

Streszczenie. *W oparciu o autonomiczne agenty programowe zdolne do obliczania indywidualnych numerycznych problemów pola, przedstawiono rozproszoną metodę rozwiązywania stanów przejściowych pola. Agenty programowe działają na zasobach rozproszonych połączonych za pośrednictwem sieci i reprezentują środowisko obliczeń dynamicznych. Komunikacja i wymiana danych między wieloma agentami umożliwia ich współpracę i pozwala podejmować decyzje w oparciu o rozproszoną wiedzę ogólną. Jako unikalna charakterystyką jest fakt, że żadna jednostka centralna nie wpływa w żadnym momencie na proces rozwiązania. Przedstawiony przykład symulacji i jej oszacowany proces obliczeniowy dowodzi, że metoda umożliwia korzystanie z nadmiarowych zasobów.*

Słowa kluczowe: automatyczna kontrola wielkości kroku, przetwarzanie rozproszone, agenty programowe, symulacja przejściowa

Introduction

Using the finite element method (FEM) to solve transient simulations is mostly done by sequentially calculating discrete time steps. The time stepping is controlled by the error

$$\mathrm{E}_{\tau}^{n} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\frac{|\epsilon_i^n|}{A_{tol,i}+\max(|u_i^{n-1}|,|u_i^n|)\cdot R_{tol,i}}\right)^2} \leq 1 \qquad (1)$$

calculated for each time step n. The number of degrees of freedom (dofs) is given as k, A_{tol} and R_{tol} represent desired tolerances, u_i^n is the approximation of the solution and ϵ_i^n is the solver's estimation of the (local) absolute error [5]. The optimal step size is

$$\Delta t_{\tau,opt}^{n} = \Delta t^{n} \cdot (1/\mathrm{E}_{\tau}^{n})^{1/(q+1)} \qquad (2)$$

with q as order of the method and Δt^n as previous step. The optimal steps size is multiplied by a safety factor f and used for calculating the next time step Δt^{n+1} [2]. If $\mathrm{E}_{\tau}^{n} \leq 1$ is satisfied the solution $\boldsymbol{u}^n$ is used and a next time step is calculated. Otherwise results are discarded and another step size is chosen.

Alternatives to a sequential calculation sequence represent parallel time integration methods proposed during the last 50 years [4]. In analogy, the developed parallel method aims on a faster calculation compared to a sequential one by performing multiple redundant shootings in parallel and selecting the largest valid steps size by evaluating error criteria like (1). Since redundant resources are provided within common business networks, they are used for calculating redundant shootings and the related partial time steps. A goal oriented usage of these additional resources is the major target of this contribution. For topics like efficiency and scalability, trust is given to the principles of the market for providing results. Within a working market, these issues are automatically evaluated by the participants and there is no need for further attention within the solution strategy.

The following contribution is organized in five sections. While section I gives the introduction for this new way of handling transient simulations, section II describes the implemented calculation framework. In section III the solution process of this framework is explained. The numerical model given in section IV proves the concepts functionality and allows its discussion based on numerical results. Section V finally concludes this contribution and outlines related topics.

1. Calculation system

Here, a distributed calculation system with independent and autonomous calculation units is used to solve transient FEM simulations. The calculation units are implemented based on the paradigm of agent based programming [7]. This enables autonomous and distributed software units to cooperate and to fulfill own goals based on individual strategies. Here, each software agents represents an independent computer with different hard- and software capabilities. This setup is shown in Fig. 1 for p agents within a computer network. Available software capabilities are FEM tools. These are provided to the agents by interfaces and enables them to handle and perform numeric field calculations. In comparison to previous work, capabilities for a transient simulation were added [8].

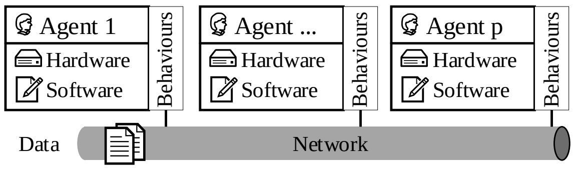

Fig. 1. Software agent system with own resources for field calculation

The agents' goal is to make profit. Therefore, agents apply their behaviours and all their capabilities accordingly. This means that the autonomous agents use their individual scheduler, their own business model and as many resources as considered necessary. It also enables a cooperative usage of the resources by multiple applications within every software agent. In the same way a competition between the agents is started to maximize their profit. Compared to cluster computing agents do not provide calculation time here. They offer individually calculated results that were derived from a global numerical model. Consequently, different solutions coexist within the calculation system. For additional processing steps like transient or coupled simulation or post-processing an adequate payment is expected by the agents. Suitable payment criteria for a user are the calculation time, the remaining error or costs for calculation like for a usage of software licences.

Enabling redundant agents to compete within the solution process of a transient numerical simulation, the agents resulting goal is to contribute to the overall solution. Therefore, they perform calculations without fulfilling distributed tasks or parallelize them in a common manner. They gain profit by quickly providing best and cheapest partial simulation results to the user before other competing agents do. For the considered transient simulation, partial tasks arise from different safety factors f for calculating a time step Δt_τ^n. These independent partial tasks are calculated in parallel. Communication between the agents avoids the calculation of duplicated parameter sets. It also enables a globally concerted uniform decision for the largest valid time step $\Delta t_{\tau,max}^n$ fulfilling (1). The iterative solution process with multiple time steps is achieved by exchanging the latest valid result.

2. Numerical method

To implement an adequate global calculation method for transient electrical engineering problems, the method is supposed to handle differential algebraic and stiff simulations [9]. Therefore, an implicit time integration scheme is preferred with a variable and adaptive time stepping based on selected error criteria. Within each time step a nonlinear system of equations is likely and must therefore been solved. The first order predictor of the implemented second order backward differential formulation (bdf) is a given in [2] as

$$\boldsymbol{u}_{pred}^n = \boldsymbol{u}^{n-1} + \frac{\boldsymbol{u}^{n-1} - \boldsymbol{u}^{n-2}}{\Delta t^{n-1}} \Delta t^n. \tag{3}$$

The bdf with a linear damping matrix D and a nonlinear stiffness matrix K^n is given as

$$K^n \boldsymbol{u}^n = -\frac{1}{\Delta t^n\, m_{bdf}^n} D\left(\boldsymbol{u}^n - \boldsymbol{u}_{bdf}^n\right). \tag{4}$$

According to [2], the parameter m_{bdf}^n is weighting the current and previous time step by

$$m_{bdf}^n = \frac{\Delta t^{n-1} + \Delta t^n}{\Delta t^{n-1} + 2\Delta t^n}. \tag{5}$$

The second order backward differential term is

$$\boldsymbol{u}_{bdf}^n = w_1^n \boldsymbol{u}^{n-1} - w_2^n \boldsymbol{u}^{n-2} \tag{6}$$

with the weighting factors

$$w_1^n = \frac{(\Delta t^{n-1} + \Delta t^n)^2}{(\Delta t^{n-1})^2 + 2\Delta t^n \Delta t^{n-1}} \tag{7}$$

and

$$w_2^n = \frac{(\Delta t^n)^2}{(\Delta t^{n-1})^2 + 2\Delta t^n \Delta t^{n-1}}. \tag{8}$$

Because of the second order accuracy of the time solution, a third order error estimator $\boldsymbol{\epsilon}$ is used. The estimator for $\boldsymbol{u}^n$ and the three previous results $\boldsymbol{u}^{n-1}$, $\boldsymbol{u}^{n-2}$, $\boldsymbol{u}^{n-3}$ is given as

$$\boldsymbol{\epsilon}^n = \frac{\Delta t^n + \Delta t^{n-1}}{6} \left[\frac{1}{\Delta t^n} (\boldsymbol{u}^n - \boldsymbol{u}^{n-1}) - \frac{\Delta t^n + \Delta t^{n-1}}{(\Delta t^{n-1})^2} (\boldsymbol{u}^{n-1} - \boldsymbol{u}^{n-2}) + \frac{\Delta t^n}{\Delta t^{n-1} \Delta t^{n-2}} (\boldsymbol{u}^{n-2} - \boldsymbol{u}^{n-3}) \right]. \tag{9}$$

As relative time discretization error (1) is used. Its absolute tolerance $A_{tol,i}$ depends on the type of the dependent variable. Especially for multiphysics systems this absolute tolerance must be chosen appropriately. Analog to (1) relative errors for the nonlinear Newton steps E_ν^n and the iterative linear solver E_μ^n are calculated. A time step Δt^n is valid if all three errors (E_ν^n, E_μ^n, E_τ^n) are within their predefined tolerances. Larger time steps during the simulation lead to large time errors E_τ^n in the numerical result, an increased number of iteration in the linear and the nonlinear solver and might end up in divergence. The optimal time step size for the nonlinear Newton method is

$$\Delta t_{\nu,opt}^n = \Delta t^n \cdot (1/\mathrm{E}_\nu^n)^{1/\mathrm{r}^2} \tag{10}$$

with r as number of Newton iterations. If the required accuracy is not reached during the linear Krylov iterations, Δt^n is repeatedly calculated with a heuristically reduced safety factor ¼ [6]. In case of E_ν^n or E_μ^n are not satisfied during the calculation of Δt^n, the time step is repeatedly calculated with

$$\Delta t_{opt}^n = \min\left(\Delta t_{\nu,opt}^n, \Delta t_{\tau,opt}^n\right) \tag{11}$$

The time step Δt_{opt}^n is also used as approximation of Δt^{n+1}. Since Δt_{opt}^n is only an approximation for Δt^{n+1} the safety factor f is used to avoid its repeated calculation [5].

For a parallel and redundant calculation of the time riod $[t_0, t_{STEP}]$ by a software agents system, each agent j applies a unique time step multiplier f_j to create and perform its simulations. The individual time step of agent j is given as

$$\Delta t_j^n = \Delta t_{opt}^{n-1} \cdot f_j. \tag{12}$$

Fig. 2 shows the resulting configuration of a software agents system. The multiplier f_j is generated based on the number of currently available and potentially contributing software agents. If only one agent is available $f = 5/6$ is chosen according to [2]. In case of more agents, larger time steps speed up the calculation. Smaller time steps avoid a re-initialization in case of no or slow convergence and require less effort during the nonlinear and the linear solver iterations. For a dynamic calculation environment it also makes sense to adapt f according to the amount of available resources after finishing the calculation of a time step. The agent with the largest time step fulfilling all error criteria proposes the solution $\boldsymbol{u}^n$ for the actual time step Δt^n. Same holds for the next optimal step size Δt_{opt}^{n+1}, so other agents are able to derive their time step by using their multiplier f_j. In case off all agents fail the new time step is chosen under the condition that all new time steps multiplied with the factor f_j are smaller than all previous steps. For a global time $t_{STEP} > t_{END}$ the calculation finally ends.

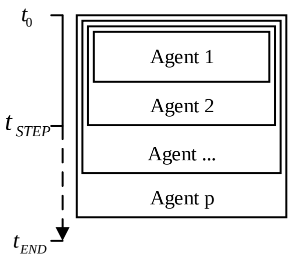

Fig. 2. Time steps selection for autonomous software agents

3. Numerical validation

To demonstrate the performance of the method a tuned circuit is considered. It contains a lumped capacitor with 1mF and an inductor with a coil of 50 thin windings including its nonlinear and lossy core. The eddy current losses and the nonlinear BH-material properties of the magnetic core are evaluated using the FEM based on the geometric 3D model shown in Fig. 3. For its calculation the sector symmetry, its horizontal symmetry and infinite elements at the border of the surrounding air are used [10].

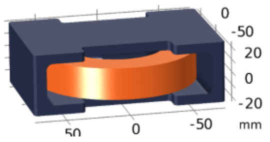

Fig. 3. Geometric model of the nonlinear and lossy inductance

The evaluated differential equation in the magnetic core is

$$\nabla \times \frac{1}{\mu(|\boldsymbol{B}|)} \boldsymbol{\nabla} \times \boldsymbol{A} = -\sigma_{core} \frac{\partial \boldsymbol{A}}{\partial t}, \tag{13}$$

with the scalar permeability μ as nonlinear function of the magnetic flux density $\boldsymbol{B} = \nabla \times \boldsymbol{A}$. Here the modified vector potential $\boldsymbol{A}$ defined by $\boldsymbol{E} = -\partial A/\partial t$ is used. The advantage of this modification is the vanishing electric scalar potential resulting in a smaller equation system [3].

For the surrounding air region and the windings $\mu_r = 1$ is considered. Within the region of the windings the additional source term $\boldsymbol{s}I/a$ is added to the right side to handle the electric current. Here $\boldsymbol{s}$ has a length of one, points in the direction of the homogenized electric current density $\boldsymbol{J}$ and normal to the cross section area a of the coil. The conductivity in the air and coil is set to $\sigma_{air} = 1\,S/m$ to provide a well posed equation system [1]. The core is modelled with an electric conductivity $\sigma_{core} = 10^5\,S/m$. The coupled ordinary differential equations are

$$\frac{\mathrm{N_W}}{a}\int_{\Omega_{coil}} \frac{\partial \boldsymbol{A}}{\partial t} \cdot \boldsymbol{s}\,\mathrm{d}\boldsymbol{x} + U = 0 \tag{14}$$

and

$$I + C\frac{\partial U}{\partial t} = 0 \tag{15}$$

with the number of windings $\mathrm{N_W}$ and the capacitor C. The introduced electric current I leads to the differential algebraic equation system

$$\begin{bmatrix} K(A) & 0 & R^T \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} \boldsymbol{A} \\ U \\ I \end{Bmatrix} + \begin{bmatrix} D & 0 & 0 \\ R & 0 & 0 \\ 0 & C & 0 \end{bmatrix}\frac{\partial}{\partial t}\begin{Bmatrix} \boldsymbol{A} \\ U \\ I \end{Bmatrix} = 0 \tag{16}$$

that also supports a higher accuracy for I during calculation. Initially the capacitor is charged with $U = 20\,V$ and the electric current I is zero. Analogy to (4) the nonlinear stiffness matrix of the FEM model is $K(A)$. Its linear damping matrix is D due to the cores conductivity. The lumped capacitor is considered as C. The integral operator $R \coloneqq \mathbb{R}^N \to \mathbb{R}$ couples $\boldsymbol{A}$ to the voltage at the capacitor. Via I as auxiliary variable the current density $\boldsymbol{J}$ is impressed into the FEM model by the operator $R^T \coloneqq \mathbb{R} \to \mathbb{R}^N$. This results in a model with $5 \cdot 10^5$ dof. One of them represents the electric current $I \in \mathbb{R}$ within the resonant circuit and another the electric voltage at the capacitor $U \in \mathbb{R}$. The other dofs are used for the magnetic vector potential $\boldsymbol{A} \in \mathbb{R}^3$ and describe the field variable.

The considered software agents system contains five agents with similar capabilities and hardware resources. They are running on five Intel i5-4690 with 4 cores at 3.5 GHz. For evaluation the time between $t_0 = 0$ and $t_{END} = 100$ ms is chosen. During the calculation process every agent performs at maximum 3 nonlinear iterations with 20 linear iterations per time step Δt^n. These are performed by the biconjugate gradient stabilized method (BiCGStab) with a vector multigrid preconditioner and a direct solver for the coarse problem. The number of Newton iterations are chosen as constant to achieve an equal distribution of the calculation effort. This also avoids idle time or time outs while waiting for partial results. The agents' time step multipliers f_j are chosen as shown in Tab. 1.

Table 1. Time steps applied by the different software agents

Agent 1	Agent 2	Agent 3	Agent 4	Agent 5
0.25	0.5	1.0	1.5	2.0

All valid voltage results of the agents are shown in Fig. 5 for four selected iterations near the minima of U. Agents that do not converge at all are not shown. The maximum time steps fulfilling all error criteria are highlighted with a circle. These results are chosen as next starting point for further time steps. The adaptive time stepping within the method is also shown in Fig. 5 by the distinct width of each iteration.

The calculated time dependent U and I are shown in Fig. 4. Calculation was done within 3 h. Additionally, reference solutions for U and I are given computed with much smaller time steps. They are evaluated on Intel Xeon E5-2630 with eight cores at 2.4 GHz within 8 h and show a good agreement.

The evaluated norm of the magnetic flux density $\boldsymbol{B}$ indicates the saturation and the skin effect within the core at $t = 2$ ms. It is shown in Fig. 6 by cutting the core along the symmetry planes.

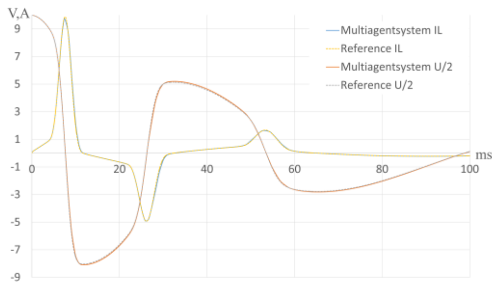

Fig. 4. Result comparison between the agents system and a reference solution

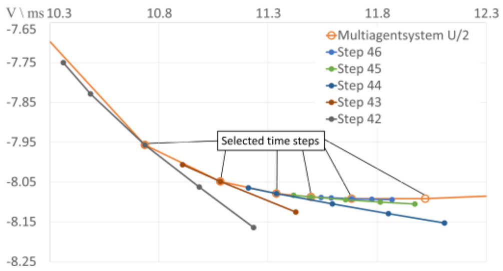

Fig. 5. Time steps of five autonomous software agents cooperatively but also competitively computing a transient FEM simulation

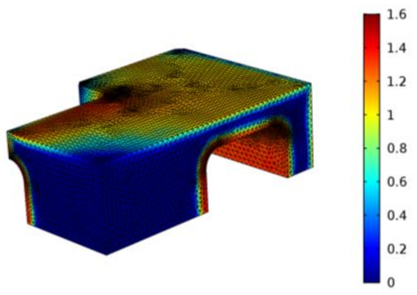

Fig. 6. Tetrahedral mesh of the iron core and norm of the magnetic flux density ***B*** *in Tesla at 2ms*

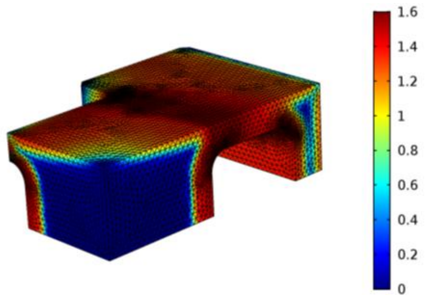

Fig. 7. Tetrahedral mesh of the iron core and norm of the magnetic flux density ***B*** *in Tesla at 5ms*

The mesh properly resolves the decay of the magnetic flux density $\boldsymbol{B}$ due to the skin effect. The magnetic flux density in the core at $t = 5$ms is shown in Fig. 7. There, the magnetic flux has penetrated the complete core and the saturation effect dominates. Due to the data exchange between the agents and their interest on contributing to further tasks all results shown in the Fig. 4–7 exist at every agent.

Here the agents system requires 145 steps to fulfill the error criteria within every time step. Compared to another run of the computation with just one agent with $f_1 = 1$ about 96 iterations are avoided. This is due to the replacement of the propose time step by results of other agents (22 times) and a totally avoided re-initialisation (74 times). The multipliers f_j considered during the solution are shown in Fig. 8. The results mostly confirm $0.5 \le f \le 1$ as advised in [2] and automatically chosen by the agents system. The usage of multiple software agents leads to a calculation time reduction of 21 %. Repeating this experiment with other error criteria the time reduction obviously relates to the number of spared iterations. Highlighting finally, that even in the worst case the presented system is as fast as a calculation without redundant resources.

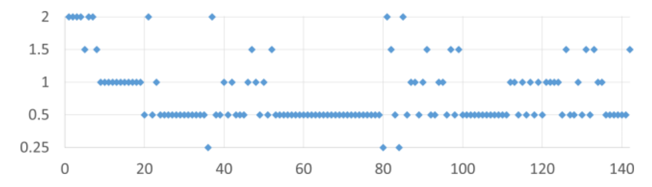

Fig. 8. Multipliers for time steps chosen to fulfill the error criteria

4. Conclusion

Within this contribution a new way of handling transient simulations is presented based on a software agents system. Details about the software agents system and the applied numerical method are given. Demonstrated on the transient simulation of a resonant circuit the method was validated. The results show that the calculation process profits from using redundant resources. Even in the worst case the presented system is as fast as a common calculation system. The additionally achieved flexibility of the calculation process is also beneficial for non-permanent calculation resources.

Acknowledgement

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under the grants RU 720/15 and WE 5312/8.

References

[1] Bíró O.: Edge element formulations of eddy current problems. Computer Methods in Applied Mechanics and Engineering 169(3-4)/1999, 391–405.
[2] Celaya E.A., Aguirrezabala J.A., Chatzipantelidis P.: Implementation of an Adaptive BDF2 Formula and Comparison with the MATLAB Ode15s. Procedia Computer Science 29/2014, 1014–1026.
[3] Emson C., Trowbridge C.: Transient 3D Eddy Currents Using Modified Magnetic Vector Potentials and Magnetic Scalar Potentials. IEEE Transactions on Magnetics 24(1)/1988, 86–98.
[4] Gander M.J.: 50 Years of Time Parallel Time Integration. Multiple Shooting and Time Domain Decomposition Methods, Cham, Springer, 2015, 69–113.
[5] Hairer E., Nørsett S.P., Wanner G.: Solving ordinary differential equations I. Nonstiff Problems, 2nd ed., Springer, Berlin 2009.
[6] Hindmarsh A.C., Brown P.N., Grant K.E., Lee S.L., Serban R., Shumaker D., Woodward C.S.: SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software 31(3)/2005, 363–396.
[7] Jennings N.R.: On agent-based software engineering. Artificial Intelligence 117/1999, 277–296.
[8] Jüttner M., Grabmaier S., Vögeli D., Rucker W.M., Göhner P.: Coupled Multiphysics Problems as Market Place for Competing Autonomous Software Agents. IEEE Transactions on Magnetics 53(6)/2017.
[9] Moore P.K., Petzold L.R.: A stepsize control strategy for stiff systems of ordinary differential equations. Applied numerical mathematics 15(4)/1994, 449–463.
[10] Zienkiewicz O.C., Emson C., Bettess P.: A novel boundary infinite element. International Journal for Numerical Methods in Engineering 19(3)/1983, 393–404.

M.Sc. Eng. Matthias Jüttner
e-mail: matthias.jüttner@ite.uni-stuttgart.de

Former academic staff at the Institute for Theory of Electrical Engineering at the University of Stuttgart. Research topics: domain decomposition methods, multiphysics simulations, software agents.

M.Sc. Sebastian Grabmaier
e-mail: sebastian.grabmaier@ite.uni-stuttgart.de

Academic staff member at the Institute for Theory of Electrical Engineering at the University of Stuttgart. Research topics: domain decomposition methods, coupled field problems.

M.Sc. Jonas Rohloff
e-mail: mail@jonas-rohloff.de

Graduated student at the University of Stuttgart. Research topics: calculation of time dependent field problems.

M.Sc. Desirée Vögeli
e-mail: desiree.voegeli@ias.uni-stuttgart.de

Academic staff member at the Institute of Industrial Automation and Software Engineering at the University of Stuttgart. Research topic: agentbased assistance systems for parallel process coordination.

Prof. Wolfgang M. Rucker
e-mail: rucker@ite.uni-stuttgart.de

Former head of the Institute for Theory of Electrical Engineering at the University of Stuttgart. Research topic: fast calculation of electromagnetic fields.

Prof. Peter Göhner
e-mail: peter.goehner@ias.uni-stuttgart.de

Former head of the Institute of Industrial Automation and Software Engineering at the University of Stuttgart. Research topic: agentbased system in industrial automation.

Prof. Michael Weyrich
e-mail: ias@ias.uni-stuttgart.de

Head of the Institute of Industrial Automation and Software Engineering at the University of Stuttgart. Research topic: methods and tools to reduce software complexity in industrial automation.

otrzymano/received: 08.12.2017 *przyjęto do druku/accepted: 11.05.2018*