ARCHIVES OF ELECTRICAL ENGINEERING VOL. 65(3), pp. 527-539 (2016)

DOI 10.1515/aee-2016-0038

Modelling of a non-linear coil with loss in iron using the Runge-Kutta methods

JOANNA KOLAŃSKA-PŁUSKA, BARBARA GROCHOWICZ

Opole University of Technology
Faculty of Electrical Engineering, Automatic Control and Informatics
Prószkowska 76, 45-758 Opole, Poland

e-mail: {j.kolanska-pluska/b.grochowicz}@po.opole.pl

(Received: 08.10.2015, revised: 02.02.2016)

Abstract: This work presents a study on dynamics of a circuit with a non-linear coil, where loss in iron is also taken into account. A coil model is derived using a state space description. The work also includes the development of an application in C# for coil dynamics examination, where the implicit RADAU IIA method of various orders is applied for the purpose of solving non-linear differential equations modelling the non-linear coil with loss in iron.

Key words: non-linear coil, Runge-Kutta methods, stiff non-linear differential equations

1. Non-linear coil

An iron-core coil has an inductance which depends on the current flowing in the coil and as a result it makes the tested coil non-linear. The circuit containing such coil does not comply with the principle of superposition, but both Kirchhoff's laws for instantaneous values are fulfilled. An equivalent circuit of such a coil is presented in Fig. 1.

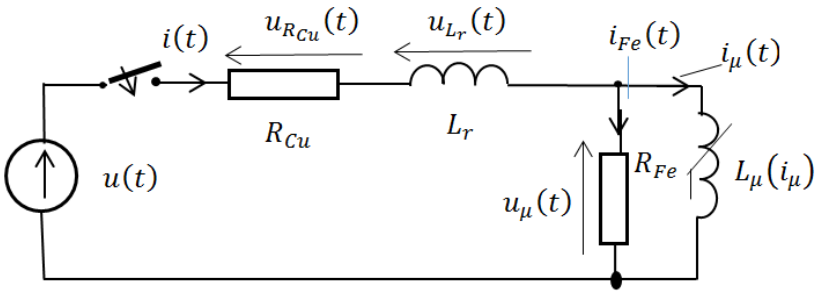

Fig. 1. Scheme of a non-linear coil with loss in iron

For the coil model from Fig. 1 the following denotations are adopted:

R_{Fe} is a resistance of equivalent circuit for the loss in iron,

R_{Cu} is a coil resistance,

L_r, L_μ are coil inductances,

U_m is a coil voltage amplitude, $u(t) = U_m \sin(\omega t + \varphi)$,

β is an initial phase supply voltage,

ω is a supply voltage ripple $\omega = 2\pi f$.

In addition to the above, the characteristics of the coil magnetic circuit, accounting for the loss in iron, is assumed to be:

$$B(H) = a_1 \operatorname{arcsh}(a_2 H) \text{ lub } H(B) = \frac{1}{a_2} \operatorname{sh}\left(\frac{B}{a_1}\right), \tag{1}$$

where B is the magnetic induction and H is the magnetic field strength.

The variables a_1, a_2 can be estimated if at least two characteristic points of magnetisation $\{B_i, H_i\}, (i = 1,2,...)$ are known. For this purpose the following non-linear system of equations will be taken into account:

$$F_i(a_1, a_2) = B_{i_1} - a_1 \operatorname{arcsh}(a_2 H_i) = 0, \ (i = 1,2,..., N_p). \tag{2}$$

In case $N_p = 2$, the set of two non-linear equations is obtained with respect to a_1, a_2, which can be solved by e.g. the Newton method. However, in the case where $N_p > 2$, the overdetermined system of non-linear equations is yielded, for which a pseudo solution is obtained with the chosen method for minimising vector norms $[F_1(a_1, a_2),..., F_{N_p}(a_1, a_2)]$.

When formulating the coil state equations, as state variables are taken the supply current $i(t)$ and the magnetic flux of the primary coil $\phi(t)$.

$$\mathbf{X}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} i(t) \\ \phi(t) \end{bmatrix}. \tag{3}$$

The average magnetic induction B in the core of the coil can be expressed by the state variable $\phi(t)$

$$B = \frac{\phi}{S_{Fe}}, \tag{4}$$

where S_{Fe} is an average cross-section of magnetic circuit.

In accordance with the magnetisation characteristic (1) it can be assumed that the average magnetic field strength in the core of the coil is as given below:

$$H = H\left(\frac{\phi}{S_{Fe}}\right). \tag{5}$$

By virtue of the flow law in the magnetic circuit of the coil:

$$H\left(\frac{\phi}{S_{Fe}}\right)l_{avFe} = wi_{\mu} = \Theta\,. \tag{6}$$

Under the second Kirchhoff's law:

$$u(t) = R_{Cu}i(t) + w\frac{\mathrm{d}\phi_c(t)}{\mathrm{d}t}, \tag{7}$$

where

$$\phi_c(t) = L_r i(t) + \phi(t)\,, \tag{8}$$

is the total magnetic flux of the coil which is the sum of the main flow $\phi(t)$ and the leakage flux $\phi_r(t) = L_r i(t)$. Substituting Equation (2) into Equation (1) gives:

$$u(t) = R_{Cu}i(t) + L_r\frac{\mathrm{d}i(t)}{\mathrm{d}t} + w\frac{\mathrm{d}\phi(t)}{\mathrm{d}t}\,, \tag{9}$$

and under the first Kirchhoff's law and the law of the current flow in form of (6) we have:

$$i_{Fe}(t) = i(t) - i_{\mu}(t) = i(t) - \frac{1}{w}H\left(\frac{\phi(t)}{S_{Fe}}\right)l_{avFe}\,. \tag{10}$$

In accordance with the coil model scheme (Fig. 1) and following the second Kirchhoff's law:

$$u_{\mu}(t) = w\frac{\mathrm{d}\phi(t)}{\mathrm{d}t} = R_{Fe}i_{Fe}(t)\,. \tag{11}$$

Substituting Equation (11) into Equation (9) the following is obtained:

$$u(t) = R_{Cu}i(t) + L_r\frac{\mathrm{d}i(t)}{\mathrm{d}t} + R_{Fe}i_{Fe}(t)\,. \tag{12}$$

Substituting formula (10) for the $i_{Fe}(t)$ current into Equations (11) and (12) we obtain:

$$\begin{aligned}\frac{\mathrm{d}i(t)}{\mathrm{d}t} &= \frac{1}{L_r}\left\{e(t) - R_{Cu}i(t) - R_{Fe}\left[i(t) - \frac{1}{w}H\left(\frac{\phi(t)}{S_{Fe}}\right)l_{avFe}\right]\right\}\\ \frac{\mathrm{d}\phi(t)}{\mathrm{d}t} &= \frac{R_{Fe}}{w}\left[i(t) - \frac{1}{w}H\left(\frac{\phi(t)}{S_{Fe}}\right)l_{avFe}\right]\end{aligned}\,. \tag{13}$$

Accounting for the selection of the state vector (3), the differential Equations (13) can be written as:

$$\begin{cases} \dfrac{\mathrm{d}x_1(t)}{\mathrm{d}t} = \dfrac{1}{L_r}\left\{e(t) - R_{Cu}x_1(t) - R_{Fe}\left[x_1(t) - \dfrac{1}{w}H\left(\dfrac{x_2(t)}{S_{Fe}}\right)l_{avFe}\right]\right\} \\ \dfrac{\mathrm{d}x_2(t)}{\mathrm{d}t} = \dfrac{R_{Fe}}{w}\left[x_1(t) - \dfrac{1}{w}H\left(\dfrac{x_2(t)}{S_{Fe}}\right)l_{avFe}\right] \end{cases} \quad (14)$$

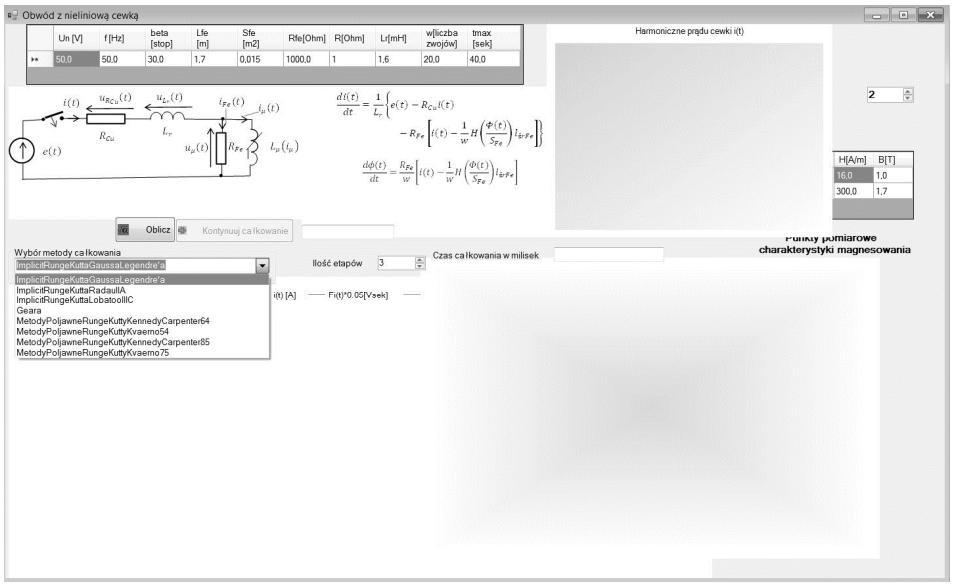

Fig. 2. Interface of the application for study on non-linear coil dynamics

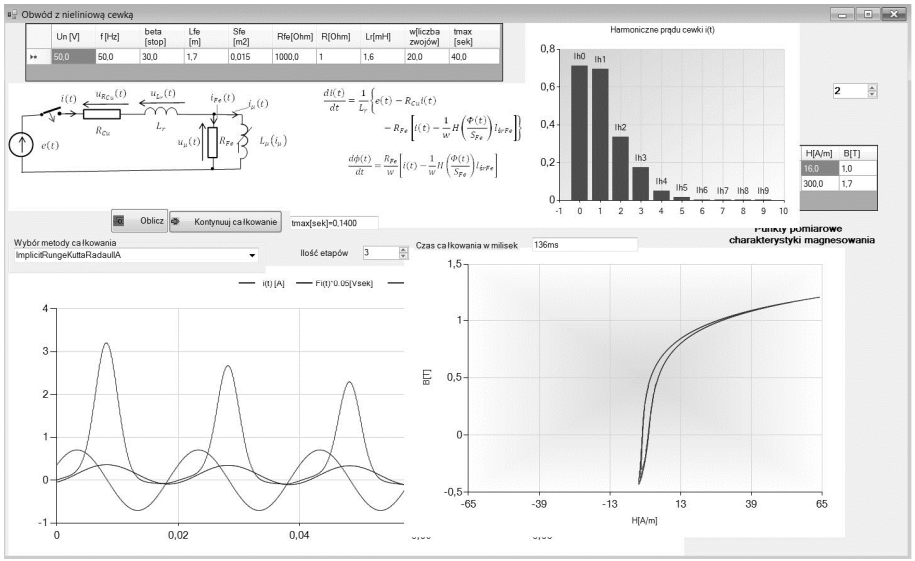

Fig. 3. Application's GUI. The characteristics of the supply voltage and the magnetizing current after 0.14 s

For the purpose of the study on non-linear coil dynamics, an application in C# was developed. The application uses the authors' library for solving differential equations using implicit RK methods [1]. The library includes basic implicit methods such as RADAU IIA, Gauss-Legendre [2] and Lobatto IIIC. The application interface is illustrated in Figs. 2, 3 and 4.

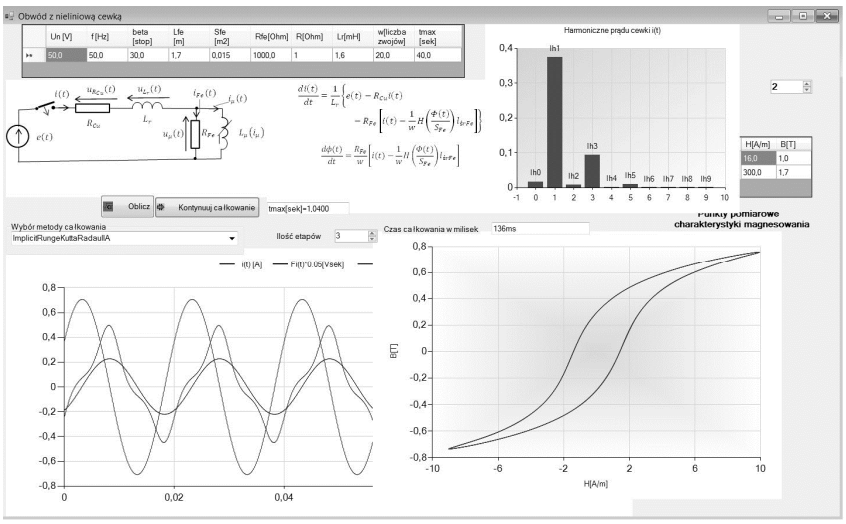

Fig. 4. The characteristics of the supply voltage and the magnetizing current after 1.14 s

2. Runge-Kutta numerical methods

Numerical solutions of differential equations for a non-linear coil with losses in iron is a well-known problem in the literature. In order to solve the system of non-linear differential equations which model the non-linear coil with iron loss, an IRK method (RADAU II A) of various orders can be successfully implemented.

2.1. Explicit RK methods (Runge-Kutta) ERK

The basis of all the Runge-Kutta methods is the integral expression

$$\mathbf{X}(t) = \mathbf{X}(t_i) + \int_{t_i}^{t} \mathbf{F}\left[\mathbf{X}(\tau), \tau\right] \mathrm{d}\tau, \;\; t > t_i, \tag{15}$$

which is equivalent to the differential Equation (16), for $t > t_i$.

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = \mathbf{F}\left[\mathbf{X}(t), t\right], \quad \mathbf{X}(t_0) = \mathbf{X}_0 . \tag{16}$$

Substituting $\tau = t_i + h\xi$, $t_{i+1} = t_i + h$ for t into Equation (15) one obtains:

$$\mathbf{X}(t_{i+1}) = \mathbf{X}(t_i) + h\int_0^1 \mathbf{F}\left[\mathbf{X}(t_i + \xi h), t_i + \xi h\right] \mathrm{d}\xi \,. \tag{17}$$

In order to determine the integrals in (17), numerous methods can be applied for the purpose of their approximations, relying on replacing them with a sum:

$$\mathbf{X}(t_{i+1}) = \mathbf{X}(t_i) + h\sum_{j=1}^{m} w_j \mathbf{F}\left[\mathbf{X}(t_i + c_j h), t_i + c_j h\right] + \mathbf{E}_m(h)\,, \tag{18}$$

where w_j and c_j are, for the given $m \geq 1$, the coefficients dependent on the assumed method for integral approximation, with $\mathbf{E}_m(h)$ being the approximation error. The Runge-Kutta methods belong into the group of the above presented method for integral approximation (18). Assuming the denotation $\mathbf{X}_i = \mathbf{X}(t_i)$, the Runge-Kutta methods can be generally described as:

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \sum_{j=1}^{m} w_j \mathbf{K}_j^{(i)}\,, \tag{19}$$

where w_j are constants.

In the class of explicit Runge-Kutta methods, the vectors $\mathbf{K}_j^{(i)}$ can be expressed as:

$$\begin{aligned} \mathbf{K}_1^{(i)} &= h_i \mathbf{F}\left[\mathbf{X}_i, t_i\right] \\ \mathbf{K}_j^{(i)} &= h_i \mathbf{F}\left[\mathbf{X}_i + \sum_{l=1}^{j-1} a_{jl} \mathbf{K}_l^{(i)}, t_i + c_j h_i\right] \quad \text{for } j > 1 \end{aligned}, \tag{20}$$

where $h_i = t_{i+1} - t_i$; $c_1 = 0$.

$$c_j = \sum_{l=1}^{j-1} a_{jl} \quad \text{for } j > 1\,. \tag{21}$$

The calculated vectors (20) used for solving the *i*-th iteration (19) can be applied in the task of discrete solution interpolation. Therefore, it is desirable to store these vectors in each iteration in form of the following $\mathbf{K}^{(i)}$ matrix:

$$\mathbf{K}^{(i)} = \begin{bmatrix} \mathbf{K}_1^{(i)} \\ \mathbf{K}_2^{(i)} \\ \vdots \\ \mathbf{K}_{Net}^{(i)} \end{bmatrix} = \begin{bmatrix} k_{11}^{(i)} & k_{12}^{(i)} & \ldots & k_{1N}^{(i)} \\ k_{21}^{(i)} & k_{22}^{(i)} & \ldots & k_{2N}^{(i)} \\ \vdots & \vdots & \vdots & \vdots \\ k_{Net1}^{(i)} & k_{Net2}^{(i)} & \ldots & k_{NetN}^{(i)} \end{bmatrix}. \tag{22}$$

The integration step is denoted by h_i instead of *h*, because by using the Runge-Kutta method it is possible to change the integration step in each iteration. For the fixed order *K* in the Runge-Kutta method, the problem relies on finding the coefficients c_j, a_{jl} and the numbers w_i such that the values $\mathbf{X}_i$ defined with relationships (19) and (20) were as close to the

exact values as possible (Table 1).

Table 1. Butcher's table

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
.	.				
c_m	a_{m1}	a_{m2}	…	$a_{m,\,m\text{-}1}$	
w^m	w_1	w_2		$w_{m\text{-}1}$	w_m

2.2. Semi-explicit Runge-Kutta methods

An important class of stiff tasks arises when solving the initial-boundary problem for partial differential equations with derivatives with respect to time. In the case when discretisation in the form of the difference quotients or the finite element method for spatial variables is applied for such equations, then as a result a large stiff sparse system of ordinary equations is obtained.

In the constructions of implicit methods, solutions with the lowest possible numerical computation cost are sought. This can be achieved not only through reduction of the approximation order but also by separation of non-linear equations. If in formula (20) we assume $a_{jl} = 0$ for $l > j$, the so-called semi-explicit ESDIRK methods *(Explicit, Singly Diagonal Implicit Runge-Kutta*) can be defined [3].

This way the system of non-linear Equations (17) whose dimension is mN will be transformed into m independent systems of non-linear equations, each of size N.

$$\text{a)}\ \mathbf{K}_1^{(i)} = h_i \mathbf{F}\left[\mathbf{X}_i + a_{11}\mathbf{K}_1^{(i)}, t_i + c_j h_l\right],$$

$$\text{b)}\ \mathbf{K}_j^{(i)} = h_i \mathbf{F}\left[\mathbf{X}_i + \sum_{l=1}^{j-1} a_{jl}\mathbf{K}_l^{(i)} + a_{jj}\mathbf{K}_j^{(i)}, t_i + c_j h_i\right] \text{ for } j = 2,.....,m\,, \tag{23}$$

$$\text{c)}\ \mathbf{X}_{i+1} = \mathbf{X}_i + \sum_{j=1}^{m} w_j \mathbf{K}_j^{(i)}\,.$$

This means that rather than solving a large system of non-linear equations, much more convenient and less computationally expensive is to sequentially calculate m systems with N unknowns in each integration step.

Bearing in mind the use of the standard algorithms for solving non-linear equations, the separated Equations (23) are generally rewritten with respect to the unknown vector $\mathbf{K}_j^{(i)}$

$$\mathbf{H}\left(\mathbf{K}_j^{(i)}\right) = \mathbf{K}_j^{(i)} - h_i \mathbf{F}\left[\mathbf{X} + a_{jj}\mathbf{K}_j^{(i)}, t\right] = 0 \text{ for } j = 1,.....,m\,, \tag{24}$$

where the calculation basis has the form:

$$\mathbf{X} = \mathbf{X}_i + \sum_{l=1}^{j-1} a_{jl}\mathbf{K}_l^{(i)}, \quad t = t_i + c_j h_i . \tag{25}$$

The system of non-linear Equations (23) is solved with the Newton method, which means that the Jacobi matrix $\mathbf{J}^{(\mathbf{H})}$ is necessary for the solution, enabling linear approximation of a vector function (of vector variable) (23).

$$\begin{aligned} \mathbf{J}^{(\mathbf{H})}\left(\mathbf{Y}^{(j)}\right)\mathrm{d}\,\mathbf{Y}^{(j)} &= \mathbf{H}\left(\mathbf{Y}^{(j)}\right) \\ \mathbf{Y}^{(j+1)} &= \mathbf{Y}^{(j)} - \mathrm{d}\,\mathbf{Y}^{(j)} \end{aligned}, \tag{26}$$

where

$$\mathbf{J}^{(\mathbf{H})}(\mathbf{Y}) = \frac{\partial \mathbf{H}(\mathbf{Y})}{\partial \mathbf{Y}} = \mathbf{1} - h_i a_{jj}\mathbf{J}^{(\mathbf{F})}\left(\mathbf{X} + a_{jj}\mathbf{K}_j^{(i)}, t\right), \tag{27}$$

$$\mathbf{J}^{(\mathbf{F})}(\mathbf{Y}, t) = \frac{\partial \mathbf{F}(\mathbf{Y}, t)}{\partial \mathbf{Y}},$$

is the Jacobi matrix of the right-hand side of differential Eq. (16), with the calculation basis

$$\mathbf{Y} = \mathbf{X}_i + \sum_{l=1}^{j-1} a_{jl}\mathbf{K}_l^{(i)}, \quad t = t_i + c_j h_i ,$$

Solving the system of linear Equations (27) within the iterative Newton process slows down the integration process in the semi-explicit Runge-Kutta methods. The larger the system of differential equations is, the more frequently it happens. In the practice of modelling of dynamical systems it very frequently occurs that the larger is the differential equation system the sparser is the Jacobi matrix $\mathbf{J}^{(\mathbf{F})}(\mathbf{X}, t)$ in the right-hand side of Equation (16). For large sparse systems, generating only non-zero elements of the Jacobi matrix can significantly reduce the time of integration. This is possible in the case when a system of ordinary differential equations would be specified not only in the form of the vector function $\mathbf{F}(\mathbf{X}, t)$ but also in the form of the matrix function

$$\mathbf{J}^{(\mathbf{F})}(\mathbf{X}, t) = \frac{\partial \mathbf{F}(\mathbf{X}, t)}{\partial \mathbf{X}},$$

which determines its non-zero elements only. Therefore in the Newton process, which is used in the ESDIRK algorithms, the methods for solving sparse linear systems should be implemented.

The recent years have brought numerous publications (e.g. [4, 5]) concerning a new subclass ESDIRK (*Explicit, Singly Diagonal Implicit Runge-Kutta*) of the SDIRK method class, in which the first stage is explicit, i.e. $a_{11} = 0$ for $c_1 = 0$. The Butcher table for this *m*-step method has the following form:

Table 2. *m*-step ESDIRK method

0	0	0	0	…	0	0
c_2	a_{21}	λ	0	…	0	0
c_3	a_{31}	a_{32}	λ	…	0	0
:	:	:	:	:	⋮	:
c_{m-1}	a_{m-11}	a_{m-12}	a_{m-13}	⋱	λ	0
1	w_1	w_2	w_3	…	w_{m-1}	λ
w	w_1	w_2	w_3	…	w_{m-1}	λ
w^*	w_1^*	w_2^*	w_3^*	…	w_{m-1}^*	w_m^*

2.3. Implicit Runge-Kutta methods

For any system of differential equations with initial conditions:

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = \mathbf{F}\left[\mathbf{X}(t), t\right], \quad \mathbf{X}(t_0) = \mathbf{X}_0, \tag{28}$$

where $\mathbf{X}(t)$ is an *N*-th dimensional vector at time *t* and $\mathbf{F}\left[\mathbf{X}(t), t\right]$ is a vector function of the vector variable $\mathbf{X}(t)$ and the time parameter *t*. The implicit Runge-Kutta methods [4-5] have the general form:

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \sum_{j=1}^{m} w_j \mathbf{K}_j^{(i)}, \tag{29}$$

where w_j are constants and the vectors $\mathbf{K}_j^{(i)}$ are as follows:

$$\mathbf{K}_j^{(i)} = h_i \mathbf{F}\left[\mathbf{X}_i + \sum_{l=1}^{m} a_{jl} \mathbf{K}_l^{(i)}, t_i + c_j h_i\right], \text{ for } j = 1,.....,m, \tag{30}$$

where $h_i = t_{i+1} - t_i$

$$c_l = \sum_{j=1}^{m} a_{jl}. \tag{31}$$

The Table of Butcher coefficients for the implicit method is as follows (Table 3).

The computing process is being carried out in base class constructors of the authors' object-oriented software.

In case of implicit methods, the coefficient vectors form a non-linear system of algebraic equations. When solving a system of *N* differential equations these vectors contain *N* components, so for the *m*-step method the system of *Nm* non-linear equations will be obtained.

Table 3. Butcher table for the implicit method

c_1	a_{11}	a_{12}	...	$a_{1,m-1}$	a_{1m}
c_2	a_{21}	a_{22}	...	$a_{2,m-1}$	a_{2m}
c_3	a_{31}	a_{32}	...	$a_{3,m-1}$	a_{3m}
...	....	...	...	...	...
c_m	a_{m1}	a_{m2}	...	$a_{m,m-1}$	a_{mm}
$\boldsymbol{w}^m$	w_1	w_2		w_{m-1}	w_m

The fact of solving the system of non-linear equations in each integration step means a relatively high cost of numerical computation.

In order to solve the system of Equations (30) the Newton method was applied [4]. Solving linear equations during the iterative Newton process undoubtedly slows down the integration process in the implicit Runge-Kutta methods. This is all the more apparent, the larger is the system of differential equations. In the practice of modelling of various systems' dynamics, it happens very often that the larger is the differential equations system the sparser is the Jacobi matrix $\mathbf{J}^{(\mathbf{F})}(\mathbf{X},t)$ of the right-hand side of equation (16).

For the implicit m-stage Runge-Kutta methods [6], there are some choices of nodes $c_1,c_2,....,c_m$ for which high orders of the methods can be obtained. As it is well known, the Gauss quadrature is of maximum approximation order. Therefore it is advisable to choose such nodes $c_1,c_2,....,c_m$ which make zeros of the quadratic formula of high order. The Gauss-Legendre, Radau and Lobatto quadratures enable to obtain the orders of the methods $2m$, $2m$-1 and $2m$-2, respectively.

In the RADAU II algorithm the nodes $c_1,c_2,....,c_m$ are the zeros of the Legendre polynomial with substitution 1-2x, which ensures that the last node is the end of the integration interval $c_m = 1$

$$P_m^{(1)}(x) \overset{def}{=} P_m(1-2x) + P_{m-1}(1-2x), \quad m = 1,2,...,s\,. \tag{32}$$

The c_j time nodes are described with the following equation:

$$P_m^{(1)}(c_j) = 0, \quad j = 1,2,...,m, \tag{33}$$

which guarantees that $c_m = 1$ so that the last time node is the node equivalent to the right end of the integration interval.

Zeros of polynomials can be estimated with a method of halving the intervals. The Vandermonde matrix $\mathbf{V}$ for these coefficients has the following form:

$$\mathbf{U}=\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ c_1 & c_2 & c_3 & \cdots & c_m \\ (c_1)^2 & (c_2)^2 & (c_3)^2 & \cdots & (c_m)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (c_1)^{m-1} & (c_2)^{m-1} & (c_3)^{m-1} & \cdots & (c_m)^{m-1} \end{bmatrix}. \tag{34}$$

Its inverse matrix $\mathbf{V}$ is now obtained:

$$\mathbf{V}=\mathbf{U}^{-1}=\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \vdots & v_{mm} \end{bmatrix}. \tag{35}$$

The knowledge of the inverse matrix $\mathbf{V}^{-1}$ enables, according to the theory of implicit Runge-Kutta methods under the simplified condition $B(m)$, to compute the coefficients w_i in the Butcher Table:

$$w_i=\sum_{k=1}^{m} v_{ik}\,\frac{1}{k},\quad i=1,2,....,\,m\,. \tag{36}$$

It turns out that the m-stage RADAU IIA method can achieve the order $2m$-1. Consider the m-stage RADAU IIA formula with the coefficients $c_1, c_2, ..., c_m$, which satisfy the Equation (18). The Butcher table [6-7] for the method of RADAU IIA for any m-stages is as below:

Table 4. Table for the 3-stage RADAU IIA method of the 5-th order

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{8296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{8296+169\sqrt{6}}{1800}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

c	$\boldsymbol{A}$
	$\boldsymbol{w}^T$

Taking into account the stability function Rz for the Runge-Kutta methods it can be proven that the RADAU II methods are A-stable [8]. This means that $Rz \le 1$ for all z for which $Rez \le 0$. Since the method's coefficients are generated with a software, Fig. 4 illustrates the RADAU IIA method's stability areas up to the 13th order.

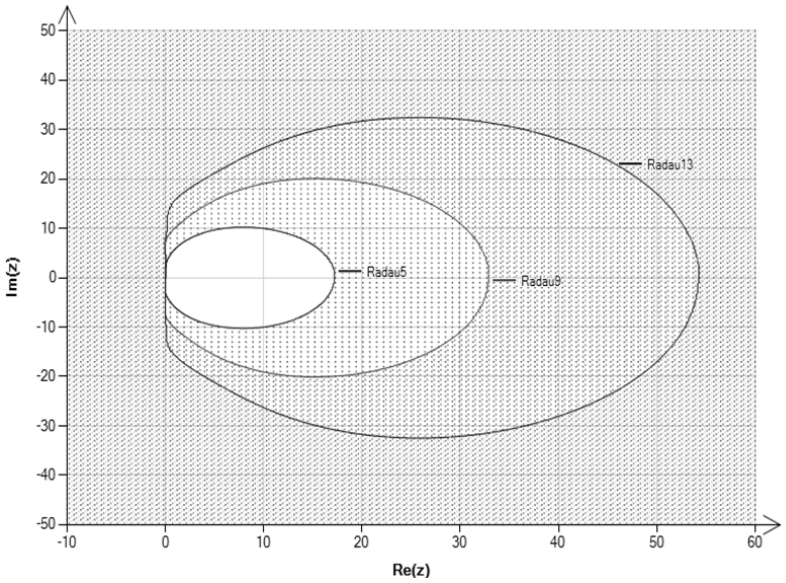

Fig. 5. RADAU IIA stability areas for orders 5, 9 and 13

3. Conclusions

Implicit IRK methods can be implemented for higher orders. However, the computation costs rapidly increase with the orders, which significantly limits their applications. In the case of using other implicit methods (RADAU IIA, Gauss-Legendre, Lobatto) the calculation cost is lower but for a large system of differential equations the computation time is also a significant barrier. As shown in the example, where the RADAU IIA method has been applied for solving the sparse equation system for a non-linear coil, the limitation to only non-zero elements of the Jacobi matrix in the iterative integration process has led to significant shortening of the integration time. Summing up, of a number of Runge-Kutta methods employed to solve specific nonlinear differential equations for the considered coil circuit, it is the RADAU IIA method that has proven to be most efficient in terms of a relatively low number of iterations and low computation times at a prespecified precision of the solution.

References

[1] Alexander R., *Design and implementation of DIRK integrators for stiff systems*, Applied Numerical Mathematics 46(1): 1-17 (2003).
[2] Kennedy C.A., Carpenter M.H., *Additive Runge-Kutta Schemes for Convection-Diffusion-Reaction Equations*, Technical report, NASA/TM-2001-211038 NASA (2001).
[3] Bijl H., Carpenter M.H, Vatsa N. Kennedy C.A., *Implicit Time Integration Schemes for the Unsteady Compressible Navier-Stokes Equations: Laminar Flow*, Journal of Computational Physics, 179: 313-329 (2002).

[4] Dormand J.R., Prince P.J., *A family of embedded Runge-Kutta formulae*, Computation Applied Maths (1980).
[5] Hairer E., Wanner G., *Solving Ordinary Differential Equations II*, *Stiff and Differential Algebraic Problems*, Springer-Verlag (1991).
[6] Jay L.O. and Braconnier T., *A parallelizable preconditioner for the iterative solution of implicit Runge-Kutta-type methods*, Journal of Computational and Applied Mathematics 111: 63-76 (1999).
[7] Butcher J.C., Chen D.J.L., *A new type of singly-implicit Runge-Kutta method*, Applied Numerical Mathematics (2000).
[8] de Swart J.B., Söderlind G., *On the construction of error estimators for implicit Runge-Kutta methods*, Journal of Computational and Applied Mathematics 86: 347-358 (1997).