Lappeenranta University of Technology
School of Energy Systems
Master`s Degree Programme in Electrical Engineering

Rostislav Teryaev

RELUCTANCE NETWORK METHOD AS ANALOGY TO FINITE ELEMENT METHOD

Examiner: Professor Olli Pyrhönen, Research Fellow D.Sc. Rafal Jastrzebski

Supervisor: Research Fellow D.Sc. Rafal Jastrzebski

ABSTRACT

Lappeenranta University of Technology
School of Energy Systems
Master`s Degree Programme in Electrical Engineering

Rostislav Teryaev

Reluctance network method as analogy to finite element method
Master's Thesis

79 pages, 27 figures, 2 tables, 21 appendices

Examiners: Professor Olli Pyrhönen, Research Fellow D.Sc. Rafal Jastrzebski

Keywords: reluctance network method, finite elements method, reluctance, nodal analysis, mesh analysis, MatLab

This study concentrates on reluctance network method implementation. The main goal of the thesis was to study algorithm of reluctance network method and its component parts. The approach for using this method for magnetic structures was developed with respect to automation. Universal procedure for circuit analysis of big circuits was implemented in MatLab. High speed of computation is reached. Validation of results showed good agreement between reluctance network method and finite elements method.

ACKNOWLEDGEMENTS

This thesis was carried out at School of Energy Systems, Lappeenranta University of Technology. I would like to express my gratitude to my supervisor D.Sc. Rafal Jastrzebski for his scientific guidance and support and also to D.Sc. Juho Montonen for his advices and thorough review work. I wish to thank the Professor Yacine Amara from University of Le Havre for his help and productive discussions. Also I express my gratitude to my friends – Evgeniy, Yuri and Maxim for listening to my stories about difficulties I faced during this process.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOLS

Chapter 2

Φ	Flux
A	Area
U	Equipotential plane or scalar potential
l	Total length of the flux tube
C	Material parameter
R	Ratio value
R_m	Reluctance
R_E	Resistance
σ	Electric conductivity
B	Magnetic flux density (induction)
H	*M*agnetic field strength
J	Current density
E	Electric field strength
F	Magnetic potential drop or flux source
U	Electric potential drop
I	Current
w	Width
h	Height
μ	Magnetic permeability
l_a	Active length
α	Angle
r	Radius

Chapter 3

1	Identity matrix
A	Matrix of nodal connections
a_{jk}	Element of matrix of nodal connections with indices j and k
memory	Size of matrix in bytes.
C	Loop matrix

c	Element of Loop matrix
$\mathbf{D}$	Sections matrix
d	Element of Sections matrix
dl	Element of the closed loop length
$d\mathbf{s}$	Element of the surface s
E	Electric field
$\mathbf{E}$	Vector of EMF source values
e	Electromotive force
$\mathbf{F}$	Substitution for $\mathbf{C}_1$
$\tilde{\mathbf{i}}$	Matrix of graph currents (One column)
i	Current
$\tilde{i}$	Generalized current
$\mathbf{J}$	Current density
$\mathbf{J}$	Vector of current source values
j	The row number
k	The column number
N	Number of different topologically not connected parts of a graph
n	Number of graph connections
n	Number of all matrix elements
nnz	Number of nonzero elements
p	Number of graph edges
q	Number of graph nodes
$\mathbf{R}$	Vector of resistance values
s	Closed surface
$\mathbf{s}$	Vector of stating nodes
$\mathbf{t}$	Vector of target nodes
$sparsity$	Sparsity of a matrix
u	Voltage drop
$\tilde{u}$	Generalized voltage drop
$\tilde{\mathbf{u}}$	Matrix of graph edges voltages (One column)
$\mathfrak{I}$	Current source current

Chapter 4

s	Vector of starting nodes
t	Vector of target nodes
R	Vector of branches resistances (reluctances)
E	Vector of branches electromotive (magnetomotive) forces
J	Vector of branches currents (fluxes)

SUBSCRIPTS

In	Inner
ind	Induced
j	The (row) index related to number of node
k	The (column) index related to graph edge
m	The (row) index related to tree edge
s	The (row) index related to connection edge
out	*O*ut/outer

ABBREVIATIONS

EMF	Electromotive force
RN	Reluctance network
RNM	Reluctance network method

1 INTRODUCTION

1.1 Background

The reluctance model (or equivalent magnetic circuit) has been the main tool for electrical machines design for the last hundred years. Typically the reluctance model had to be constructed only for the main flux path. It was necessary because all computations were performed only manually those days. Moreover it assures high speed of the solution.

There is another tool to model electrical machines, which is Finite Elements Method (FEM). It is a numerical method which allows to calculate fields of any kind with some discretization. The size of discretization depends on the size of the finite element mesh. This method results into set of values of field potentials in mesh nodes. Unfortunately, it requires to perform many similar mathematical operations in a row. Finite Element method is highly precise and allows to overcome some issue which some other methods cannot, for example issues related to computation of complicated geometrical structures. However, the main disadvantage of this method is a huge number of mathematical operations which is almost impossible to perform manually.

The development of computers started in the 1980's allowed to use Finite Elements method in practice making time for computing quite acceptable, but still not really fast. Assuming above, FEM is reasonable to use in the final stages of design of electrical machines to final validations of a model or its refinement. And it is rather difficult to use FEM in pre-design stages because a structure of machine can vary in many places which will require to recompute model every time. Moreover, not only parameters values of the machine have to be changed every time, but a model geometry also.

It should be noticed that there are a number of many different approaches which can be used for engineering analysis also. They can be divided into three groups:
Analytical models [1, 2, 3, 4];
Numerical models [4, 5, 6, 7];
Semi-numerical, semi-analytical or hybrid models [8, 9, 10, 11, 12, 13].

Many of these approaches are different combinations of FEM, reluctance network method (RNM) and formal solution of Maxwell’s equations.

1.2 Goals and delimitations

The goal of this thesis was:

1. To develop a tool capable to construct automatically a reluctance network for structures with different geometrical parameters
2. Analyze this network
3. Get radial and tangential flux density component in the air-gap region
4. Validate radial flux component with one computed by FEM program

1.3 Structure of the thesis

This thesis consists of four chapters.

The Chapter 1 is introduction, which shortly describes existing methods for electrical machine analysis and goal of this paper.

In Chapter 2, meshed based reluctance network theory discussed. The basic element of the reluctance network is introduced and equations for its parts.

In Chapter 3, the circuit analysis procedure is introduced. The circuit representation by graph and matrices is given. Algorithm of solution procedure is described.

In Chapter 4, the developed procedure is applied to linear PM structure. Result is compared to reference result by FEM.

2 RELUCTANCE NETWORK

2.1 Analogies between electric and magnetic circuits

Flux tube – is a space with a quasi-stationary magnetic (or electric) field, in which (Fig. 1.) [15]:

1. All flux lines are perpendicular to their bases – which are equipotential planes,
2. No lines of flux cut their sides.

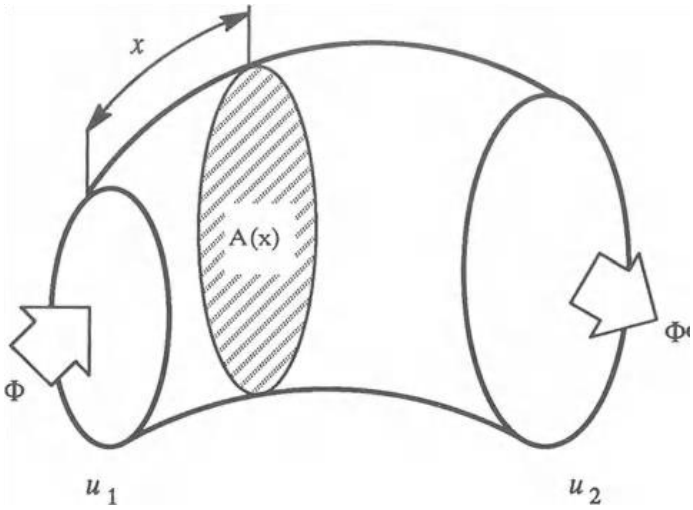

Fig. 1. A flux tube in a field [15]

Numerically flux tube can be characterized by the ratio of potential difference (Magnetic or electric) at the ends of the tube with no current and its flux. Mathematically, this ratio is equal to:

$$R = \int_0^l \frac{1}{C(x) \cdot A(x)} dx \tag{1}$$

Where

l – Total length of the flux tube

$A(x)$ – Cross-section of the flux tube

$C(x)$ – Material parameter

R – Ratio value, which has different names for different filed types

For magnetic field equations above goes as follows

$$R_m = \int_0^l \frac{1}{\mu(x) \cdot A(x)} dx \tag{2}$$

Where

μ – Magnetic permeability

R_m – Reluctance

For electric field

$$R_E = \int_0^l \frac{1}{\sigma(x) \cdot A(x)} dx \tag{3}$$

Where

R_E – Resistance

σ – Electric conductivity

Relationship between flux density and field strength for magnetic and electric fields are:

$$B = \mu H \tag{4}$$

$$J = \sigma E \tag{5}$$

Assuming above table 1 can be written

Table 1. Analogies between magnetic and electric fields

	Type of field	
	Magnetic	Electric
Material constant	μ	σ
Flux or current density	B	J
Field strength	H	E
Potential difference	$F = \int_0^l Hdl$	$U = \int_0^l Edl$
Flux or current	$\Phi = \int BdA = \frac{F}{R_m}$	$I = \int JdA = \frac{U}{R_E}$
Reluctance or Resistance	$R_m = \int_0^l \frac{dx}{\mu(x) \cdot A(x)}$	$R_E = \int_0^l \frac{dx}{\sigma(x) \cdot A(x)}$

By comparing expressions for magnetic and electric fields the analogies can be noticed. They are based on the same mathematical principles but with different names of parameters. It allows to construct a magnetic equivalent circuit that is analogous to a resistive electrical circuit.

2.2 Reluctance as analogy to resistance

Equation (2) can be expressed in different ways for different shapes of flux tubes. However, we will consider three most common kind of shapes, which are used in practice [11].

The radial and circumferential reluctance components for the rectangle (Fig. 2.):

$$\begin{cases} R_{mr} = \dfrac{1}{\mu}\dfrac{1}{l_a}\dfrac{h}{w} \\ R_{mc} = \dfrac{1}{\mu}\dfrac{1}{l_a}\dfrac{w}{h} \end{cases} \tag{6}$$

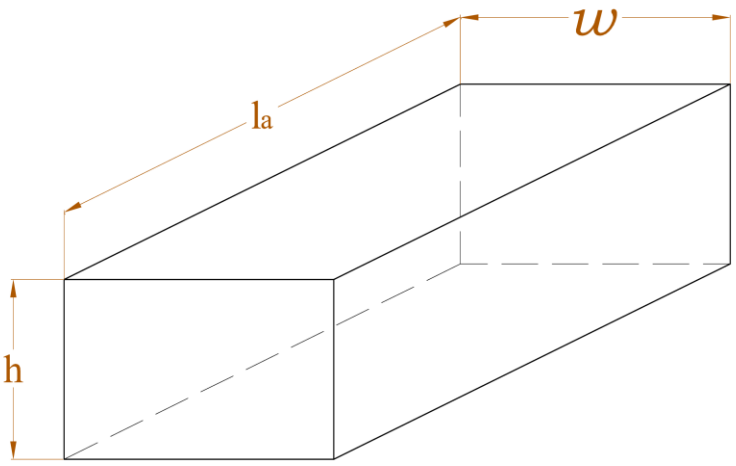

Fig. 2. Rectangular flux tube

The radial and circumferential reluctance components for the trapezium (Fig. 3.):

$$\begin{cases} R_{nr} = \dfrac{1}{\mu}\dfrac{1}{l_a}\left[\dfrac{h}{(w_2 - w_1)}\ln\left(\dfrac{w_2}{w_1}\right)\right] \\ R_{nc} = \dfrac{1}{\mu}\dfrac{1}{l_a}\left[\dfrac{h}{(w_2 - w_1)}\ln\left(\dfrac{w_2}{w_1}\right)\right]^{-1} \end{cases} \tag{7}$$

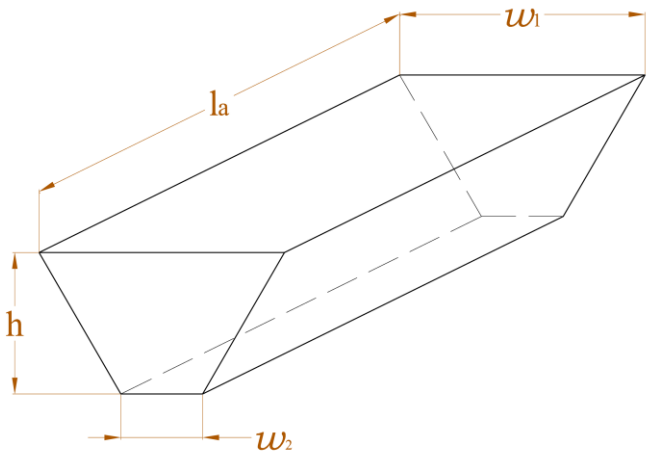

Fig. 3. Trapezoidal flux tube

The radial and circumferential reluctance components for the sector (Fig. 4.):

$$\begin{cases} R_{mr} = \dfrac{1}{\mu}\dfrac{1}{l_a}\left[\alpha \Big/ \ln\left(\dfrac{r_{out}}{r_{in}}\right)\right]^{-1} \\ R_{mc} = \dfrac{1}{\mu}\dfrac{1}{l_a}\left[\alpha \Big/ \ln\left(\dfrac{r_{out}}{r_{in}}\right)\right] \end{cases} \tag{8}$$

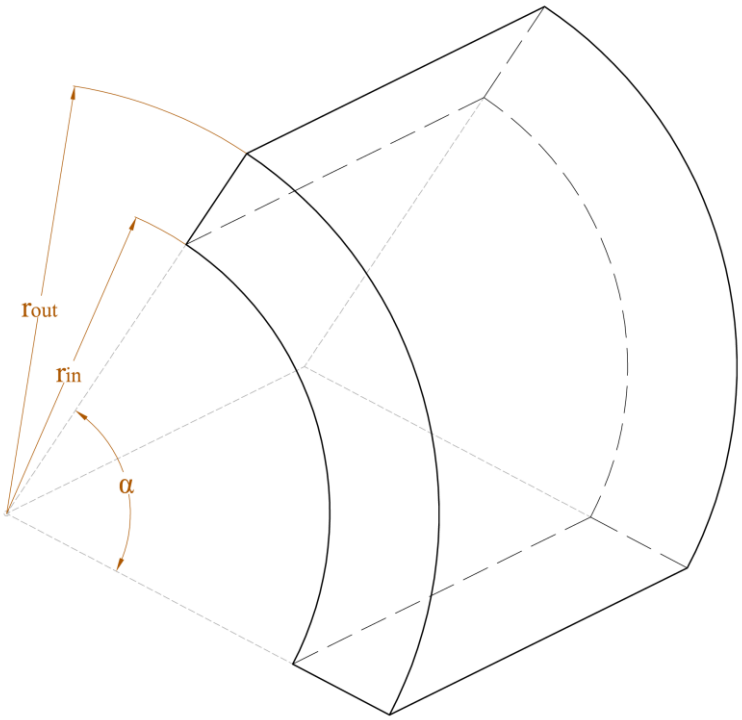

Fig. 4. Sector flux tube

2.3 Flux source or magnetomotive force as analogy to current source or electromotive force

Every energy source can be presented in two ways which are flux source in parallel to reluctance element or magnetomotive force in series with reluctance [14].

Flux source (Fig. 5.):

Equations:

$$\begin{cases} U_2 - U_1 = R_m \cdot (\Phi - \Phi_s) \\ U_2 - U_1 = R_m \cdot \Phi_m \end{cases}$$

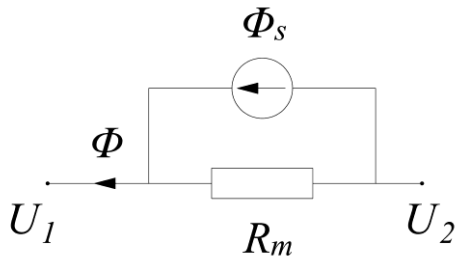

Fig. 5. Flux source

Magnetomotive force (Fig. 6.):

Equations:

$$\begin{cases} U_2 - U_1 = R_m \cdot \Phi - F \\ U_2 - U_1 = R_m \cdot \Phi_m \end{cases}$$

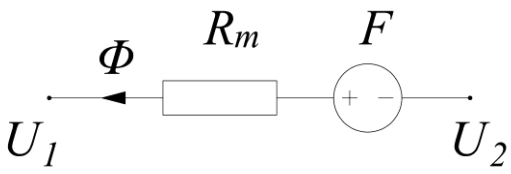

Fig. 6. Magnetomotive force

Where, assuming the shape of reluctance element as, for example, rectangle Φ or F can be found as follows (v and h indices denotes vertical and horizontal elements)

For vertical direction of the magnetic flux density B (Same as on Fig. 7.)

$$\begin{cases} \Phi_{s\_v} = B \cdot (w \cdot l_a) \\ \Phi_{s\_h} = 0 \end{cases}$$

For horizontal direction of the magnetic flux density B

$$\begin{cases} \Phi_{s\_v} = 0 \\ \Phi_{s\_h} = B \cdot (h \cdot l_a) \end{cases}$$

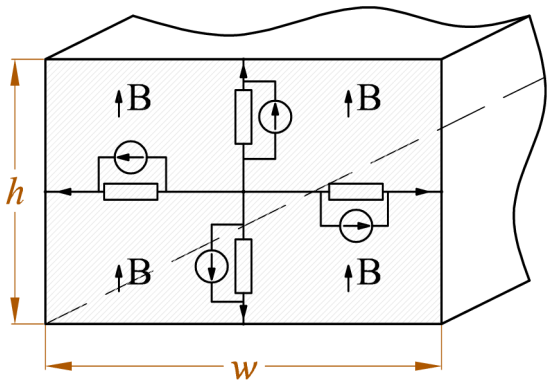

Fig. 7. Flux source

For any direction of the current density J

$$\begin{cases} F_v = J \cdot \left(\frac{h}{2}\right) \cdot w \\ F_h = J \cdot h \cdot \left(\frac{w}{2}\right) \end{cases} \Rightarrow F = J \cdot \frac{h \cdot w}{2}$$

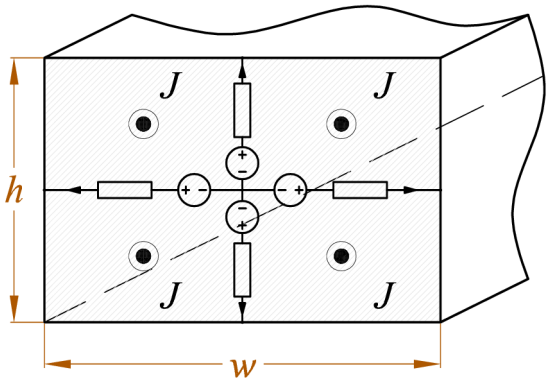

Fig. 8. Magnetomotive force

After all elements have been defined it is possible to create a reluctance network for a structure in study. However, it is not as straightforward as it seems to be.

Reluctance network is a magnetic circuit, which has to be formed before analyzing. Forming this circuit requires two stages:

1. Topological structure of this network has to be set, in other words it is necessary to describe all branch connections and set this information to computer for further processing.
2. Values of all reluctances and sources also have to be set.

To do these actions for a small circuit is an easy task: topology structure is easy enough, so circuit can be analyzed manually. Number of reluctance elements is also small, thus its values can be set manually again. However, it is necessary to have many values of magnetic potential in particular areas, for example, in air-gap region to get proper plots. Therefore, reluctance network will be a big circuit. For big circuits completing mentioned stages is far more complicated task.

1. To analyze big circuit manually will take infinite amount of time, thus it is necessary to use a method, which is capable to solve circuits on a computer. There are several of these methods. Thanks to Kirchhoff's laws it is possible to convert a problem of analyzing a circuit to a problem of solving system of liner equations. Algorithms for solving linear equations are well developed. Next chapter contain an in-depth description of the approach and features of its implementation in MatLab.
2. However, there is no systematic method to work with the task of setting values to reluctance elements. In general, sizes are different for all reluctance elements in

case of a structure with a complicated shape. Therefore, it is necessary to separately set values for all elements manually. For big circuits this task is even more complicated when solving the circuit by hand. However, it is possible to simplify this stage by assuming same size for all elements. This approach is used in [14]. This approach has its constraints, but it is necessary for partial automation of the problem. Thus, it is necessary to assign only indices to all reluctance elements, which describe the type of material. These indices can be easily converted to magnetic permeability and reluctance is calculated for the elements. However, it is still not easy to assign material indices to all elements in case of a big network. Chapter 4 describes how this problem was simplified by the author of this paper.

3 CIRCUIT ANALYSIS

3.1 Theory block

This section describes theory which is laid behind two main methods for analysis of electrical circuits and their matrix representation. In spite of magnetic nature of our problem this section operates with electrical terms because described methods are presented in literature applying to electrical circuits [16, 18, 19]. However, due to analogy between electric and magnetic circuit [15, 17] it can be applied to magnetic problems as well. This section is mainly based on [16], where authors describe matrix and graph methods related to circuit theory what was first developed by Gustav Kirchhoff.

3.1.1 Topological terms of electrical circuit diagram. Circuit diagram graph

Nodes in circuit diagrams are depicted by dots. In complicated diagrams, where mutual line crossing is possible for depicting the existence of their connections dots are also used (for example, dots *a, b, c, d, e, g, g, a', b', c'* on Fig. 9.). Formally, all these dots can be considered as diagram nodes. These imaginary nodes are special in that way that they are connected by branches with current flowing through them and have no voltages as resistance of these branches is considered zero. Therefore, potentials of these nodes are equal and can be depicted as one node. On Fig. 9. nodes *a*, *b*, *c* can be combined to the one node, *a', b', c'* – to another, *d*, *e*, *f*, *g* – to the third one (Fig. 10.).

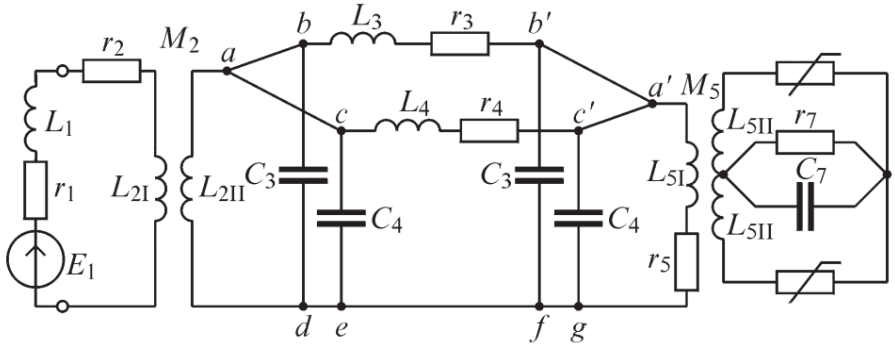

Fig. 9. Demonstration of term dot for depicting circuit nodes [16]

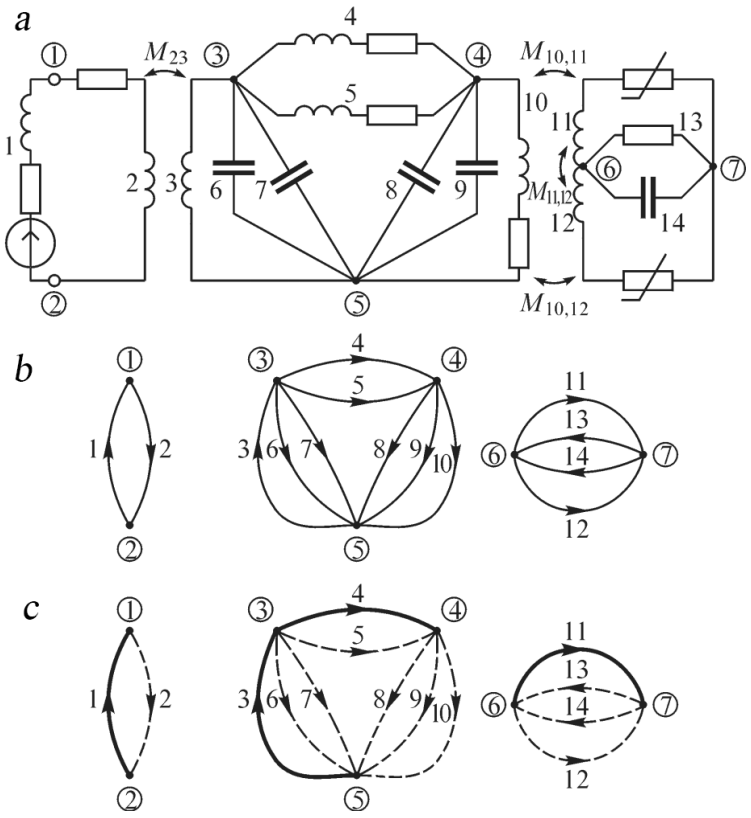

Fig. 10. Circuit diagram and its graph [16]

To understand mutual branch connections in a more clear way it is considered that branches are presented as lines – graph edges and nodes as dots – graph nodes. This topological representation of electrical circuit diagram is called *graph of the electric circuit* or shorter – *circuit graph.*

It has to be noticed that current and electromotive force (EMF) sources are not presented on a topological diagram. However, branch with EMF source remains but ones with current sources does not because its inner resistance is infinite.

Graph is called *connected* when there is a path between any pair of vertices.

Graph is called *directed* if every edge of this graph has a direction associated with this edge.

Directed graph of diagram (Fig. 10. *a*) is presented on Fig. 10. *b*. It can be noticed that because of mutual inductance the graph of the diagram is split into three different unconnected parts.

Let us agree to number nodes as numbers in circles, but edges as numbers without circles. On diagram graph (Fig. 10. *b*) there are 7 nodes and 14 edges.

One of the important terms related to graph is a *graph tree*, which is a set of branches (which also comprise a graph) connecting all nodes of the initial graph without loops. One graph can have many trees. Consider to depict edges comprising a tree as bold lines.

Edges which complete a graph tree to the full graph and consequently not being a parts of the tree are called *graph connections*. Let us agree to depict such edges as dashed or thin lines. Obviously, any graph tree has its own set of graph connections which is called co-graph. For example, for graph on Fig. 11. *b* co-graph is comprised by edges which connect nodes *ab, bf, fd, da, fa* and *bd*. One of several possible graph trees is depicted on Fig. 10. .

The electrical circuit diagram is depicted on Fig. 11. *a* with two different trees of its graph (*b, c*). If connected graph has p edges and q nodes, then its tree will have $(q-1)$ edges and $n = p-(q-1)$ graph connections. These statements are made by taking into account the definition of tree and graph connection terms, which mean that q nodes can be connected minimum by $(q-1)$ tree edges and graph connections are all remaining edges which are $p-(q-1)$. For example, in separate parts of unconnected graph (Fig. 10.) we have: in the left part of graph $p=2,\ q=2,\ n=2-(2-1)=1$; in the middle part $p=8,\ q=3,\ n=8-(3-1)=6$; in the right part $p=4,\ q=2,\ n=4-(2-1)=3$.

It is important to notice that it is not possible to state the same for the full unconnected graph. The full graph has $p=14$, $q=7$ and $n=14-(7-1)=8$ while number of graph connections is $1+6+3=10$. For unconnected graphs the number of graph connections can be get as follows $n=p-(q-1)+N-1=p-q+N$, where N – is a number of different topologically not connected parts of a graph. In our case $N=3$, therefore, $n=14-(7-1)+3-1=10$. For circuit diagram on Fig. 10. we have $p=10$, $q=5$ and $n=6$.

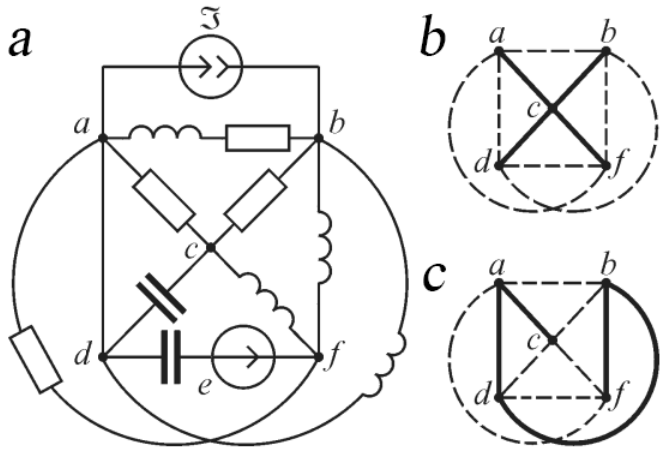

Fig. 11. Circuit, its graph and some of its trees [16]

3.1.2 Matrix of nodal connections

Depicting a circuit diagram as graph gives the possibility to present them as a table. This table has *q* rows and *p* columns, which are numbers of graph nodes and edges correspondingly. Let us agree to number rows accordingly to nodes number and columns to edges ones. The cell is described by double index *(j, k).* The first index is the row number, the second – the column number. Let fill the table by assuming following rules.

1. The cell *jk* has the value +1 if kth edge is connected to *j*-th node and the arrow is directed from jth node.
2. The cell *jk* has the value -1 if kth edge is connected to *j*-th node and the arrow is directed to jth node.
3. The cell *jk* has no value if kth edge is not connected to *j*-th node.

By following these rules the table for circuit diagram on Fig. 12. *a* and his graph on Fig. 12. *b, c* can be presented as a table

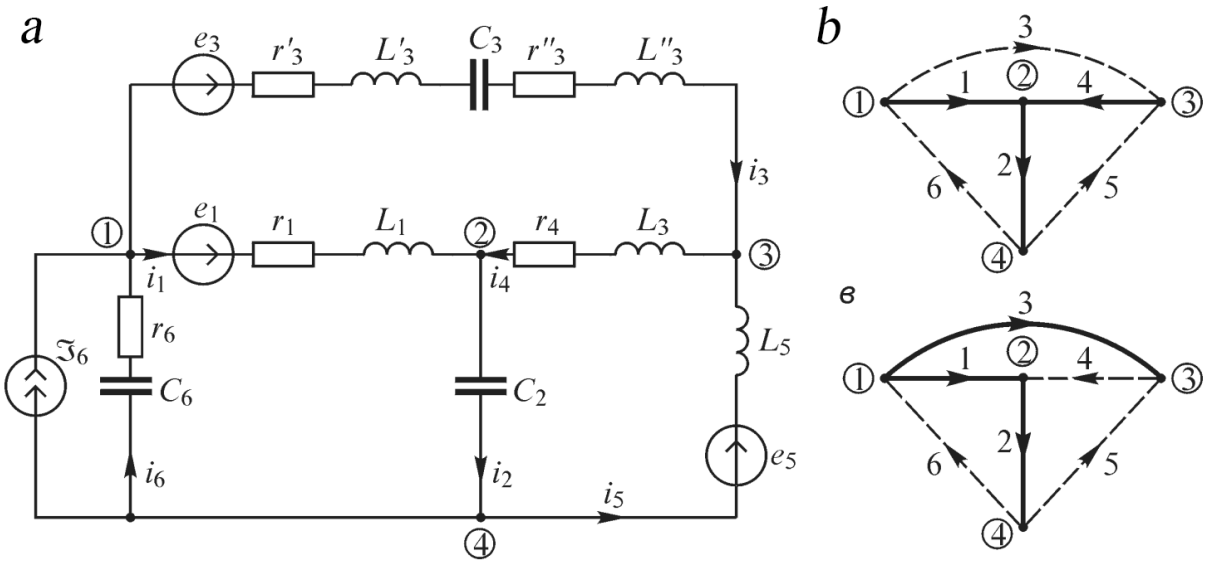

Fig. 12. Circuit, its graph and some of its trees [16]

Table 2. Table of coefficients

	1	2	3	4	5	6
1	1		1			–1
2	–1	1		–1		
3			–1	1	–1	
4		–1			1	1

Number of rows printed bold are corresponding nodes numbers.

Let us note some characteristics of this table. Each column can have only two nonzero (not empty) cells because one edge can be connected only to two nodes. Sum of one column values is zero because the arrow of every edge is directed from one node to another therefore one cell has the value +1 the another one -1. Assuming the previous fact only $(q-1)$ rows can be filled because *q*-th row always can be derived from others. This table of connections can be written as a matrix.

Matrix of nodal connections – is a rectangle matrix rows of which are correspond to nodes without one and columns to edges of directed graph of the electrical circuit diagram. Elements of this matrix are 0, +1 and -1 if the particular branch is correspondingly not connected with, directed from or directed to particular node.

Let us denote a matrix of nodal connections with bold **A.** For graph in Fig. 12. , we have

$$\mathbf{A} = \left|a_{jk}\right| = \begin{vmatrix} 1 & 0 & 1 & 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & -1 & 0 & 0 & = 2 \\ 0 & 0 & -1 & 1 & -1 & 0 & 3 \end{vmatrix}$$

1	2	3	4	5	6
1		1			−1
−1	1		−1		
		−1	1	−1	

$q-1$ rows (9)

p columns

Matrix of nodal connections has dimensions $(q-1)\times p$ which are number of rows $(q-1)$ and number of columns (p) of matrix.

3.1.3 Kirchhoff's laws

The circuit analysis is based on two Kirchhoff's laws. Let us consider them to the circuit with lumped parameters.

The first Kirchhoff's law or Kirchhoff's law for nodes (currents) is based on charge conservation principle. Let us enclose one node by closed surface *s* (Fig. 13.). Because of consumption of absence of displacement current all electricity capacity in a circuit with lumped parameters is due to capacitors connected in it. Therefore only conventional currents in conductors flow through closed surface *s*.

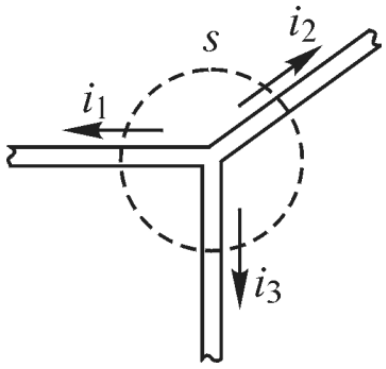

Fig. 13. Circuit node and enclosing surface [16]

In according with charge conservation principle we have

$$\oint_s \mathbf{J}d\mathbf{s} = i_1 + i_2 + i_3 = 0 \qquad (10)$$

For any number *n* of branches we have

$$\sum_{k=1}^{n} i_k = 0 \tag{11}$$

This mean that *the sum of currents flow from the node of the electrical circuit is equal to zero*. This is a formulation of the Kirchhoff's current law.

For forming equations in accordance with Kirchhoff's current law positive branch directions have to be assumed. In the left part of the equations "plus" sign has to be employed if the current flows from the node and "minus" sign if the current flows to the node. For the case depicted on the Fig. 14. one can write Kirchhoff's current law as follows

$$-i_1 + i_2 + i_3 = 0 \tag{12}$$

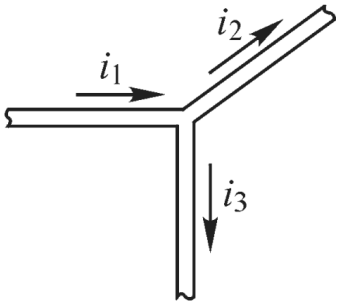

Fig. 14. Circuit node [16]

If after some calculation the result value of some current is positive $(i_k > 0)$ this means that current has the same direction as the direction of the arrow, and if the result value is negative $(i_k < 0)$ this mean that current and arrow have opposite directions.

The second Kirchhoff's law or Kirchhoff's law for loops (voltages) is applied to electrical circuit loops. For some closed loop we have

$$\oint Edl = \oint E_{ind} dl + \oint E_{out} dl \tag{13}$$

Where on the left side voltage drop on all elements and on the right side EMF from induced and outer sources (for example electromagnetic generator and accumulator correspondingly). For the circuit with lumped elements we can write previous equation as follows

$$\sum_{k=1}^{k=n} u_k = \sum_{k=1}^{k=n} e_{k\,ind} + \sum_{k=1}^{k=n} e_{k\,out} \quad or \quad \sum_{k=1}^{k=n} u_k = \sum_{k=1}^{k=n} e_k \tag{14}$$

where:

$\sum_{k=1}^{k=n} u_k$ - the sum of voltage drops on all electrical circuit elements (for example resistors, inductances, capacitors);

$\sum_{k=1}^{k=n} e_k$ - the sum of all EMF from all sources in electrical circuit.

On the basis of the above, second Kirchhoff's law claims that sum of voltage drops in all branches of any closed loop of electrical circuit is equal to the sum of all EMF sources in this loop.

3.1.4 Nodal equations for circuit currents

For electrical circuit with q nodes q equations can be written, applying to them Kirchhoff's current law. However, only $q-1$ of them are independent from each other. Independence is based on the following fact. It is always possible to find such an order of writing down these nodes that every next node will differ from the previous ones at least by one new branch. Let us note that the sum of any $j\left(1 \le j \le q-1\right)$ equations is the equation for such closed surface that enclosing these j nodes. It follows from that currents of branches which are inside the surface but does not go through it are considered in equations twice: the first one with minus sign, the second – with plus sign. For example, sum of equations for nodes *1, 2, 3, 4* and *5* of graph on Fig. 15. will define sum of currents for the surface which enclose these nodes (dashed line depicts this surface boundary).

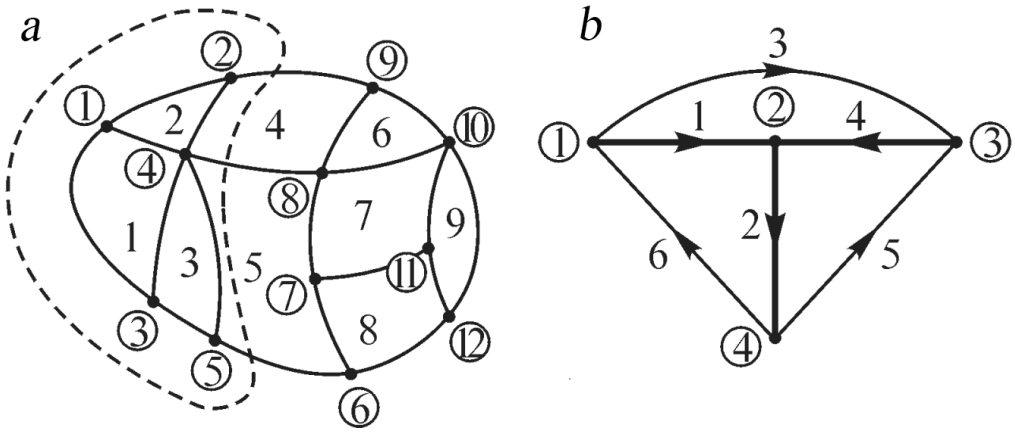

Fig. 15. Circuit and its part graph [16]

If the surface enclosing $q-1$ nodes then the sum of currents flows into this surface is equal with minus sign to the sum of currents for *q*-th node and therefore *q*-th equation is a derivative of previous $(q-1)$ equations. With respect to that we will say that circuit (or diagram graph) with *q* nodes has only $(q-1)$ independent nodes.

Depending on the direction of current to the surface normal it can be included in equation with different signs. By considering this Kirchhoff's current law can be written as follows

$$\sum_{k=1}^{p} a_{jk}\tilde{i}_k = 0, \quad j = 1...(q-1), \tag{15}$$

where $a_{jk} = \pm 1$ or $a_{jk} = 0$.

Let the normal to a closed surface is directed to outer space. If the current of *k* branch directed from node *j*, then it will be included in equation with «plus» sign, otherwise – with «minus» sign. In the first case $a_{jk} = 1$, in the second one $a_{jk} = -1$. If the *k*-th branch is not connected to this node *j*, then $a_{jk} = 0$.

In view of the above, for example, for graph on Fig. 15. *b* (graph of the circuit diagram on Fig. 12.), where $q = 4$ the system of three independent equations can be written

$$\begin{array}{lllll} \text{for node 1} & \tilde{i}_1 + \tilde{i}_3 - \tilde{i}_6 = 0, & a_{11} = 1, & a_{13} = 1, & a_{16} = -1; \\ \text{for node 2} & -\tilde{i}_1 + \tilde{i}_2 - \tilde{i}_4 = 0, & a_{21} = -1, & a_{22} = 1, & a_{24} = -1; \\ \text{for node 1} & -\tilde{i}_3 + \tilde{i}_4 - \tilde{i}_5 = 0, & a_{33} = -1, & a_{33} = 1, & a_{35} = -1. \end{array} \tag{16}$$

It can be noticed that rules for assigning signs for a_{jk} are the same as we used earlier for matrix of nodal connections.

Due to consistent approach used to assign sign for coefficients of nodal connection matrix and currents in equations we can use matrix of nodal connections for algebraization of equations of the Kirchhoff's current law.

Let us write currents in the graph (or in the circuit diagram) as a matrix with one column and *p* rows:

$$\tilde{\mathbf{i}} = \left|\tilde{i}_k\right| = \begin{vmatrix} \tilde{i}_1 \\ \vdots \\ \tilde{i}_p \end{vmatrix}, \; k = 1 \ldots p. \tag{17}$$

Every row of nodal connections matrix is a set of coefficients for currents in a Kirchhoff's first law for the node which is the same with the row number. Therefore with respect to linear algebra rules every nodal equation can be presented as follows

$$j \begin{bmatrix} a_{j1} \end{bmatrix} \; \ldots \; \begin{bmatrix} a_{jk} \end{bmatrix} \; \ldots \; \begin{bmatrix} a_{jp} \end{bmatrix} \times \begin{bmatrix} \tilde{i}_1 \\ \vdots \\ \tilde{i}_k \\ \vdots \\ \tilde{i}_p \end{bmatrix} = a_{j1}\tilde{i}_1 + \ldots + a_{jk}\tilde{i}_k + \ldots + a_{jp}\tilde{i}_p \quad \sum_{k=1}^{p} a_{jk}\tilde{i}_k = 0 \tag{18}$$

vector $(1 \times p)$ — matrix (1×1)

These matrix equations can be written for all $(q-1)$ rows of nodal connections matrix. In matrix form system of these equations is written as follows

$$\mathbf{A}\tilde{\mathbf{i}} = \mathbf{0} \; (19)$$

For circuit diagram graph (Fig. 12.) we have

$$\mathbf{A}\tilde{\mathbf{i}} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 1 & & 1 & & & -1 \\ -1 & 1 & & -1 & & \\ & & -1 & 1 & -1 & \end{bmatrix} \\ (3 \times 6) \end{array} \times \begin{array}{c} \begin{bmatrix} \tilde{i}_1 \\ \tilde{i}_2 \\ \tilde{i}_3 \\ \tilde{i}_4 \\ \tilde{i}_5 \\ \tilde{i}_6 \end{bmatrix} \\ (6 \times 1) \end{array} = \begin{array}{c} \begin{bmatrix} \tilde{i}_1 + \tilde{i}_3 - \tilde{i}_6 \\ -\tilde{i}_1 + \tilde{i}_2 - \tilde{i}_4 \\ -\tilde{i}_3 + \tilde{i}_4 - \tilde{i}_5 \end{bmatrix} \\ (3 \times 1) \end{array} = \begin{array}{c} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ (3 \times 1) \end{array} = \mathbf{0} \tag{20}$$

Every row of matrix multiplication $\mathbf{A}\tilde{\mathbf{i}}$ determines the equation for the node as stated by Kirchhoff's current law.

For *k*-th generalized branch the following equation can be written

$$\tilde{i}_k = i_k + \Im_k, \tag{21}$$

where i_k - current in passive elements of *k* branch, and $\Im_k$ -current of current source in *k* branch if they are directed from the same node.

Matrix form of system equations according to Kirchhoff's current law for currents in circuit elements can be written as follows

$$\mathbf{A}\tilde{\mathbf{i}} = \mathbf{A}\mathbf{i} + \mathbf{A}\Im = \mathbf{0}, \tag{22}$$

where

$$\underset{(p\times 1)}{\tilde{\mathbf{i}} = \begin{vmatrix} \tilde{i}_1 \\ \vdots \\ \tilde{i}_p \end{vmatrix}}; \quad \underset{(p\times 1)}{i = \begin{vmatrix} i_1 \\ \vdots \\ i_p \end{vmatrix}}; \quad \underset{(p\times 1)}{\Im = \begin{vmatrix} \Im_1 \\ \vdots \\ \Im_p \end{vmatrix}}; \quad \underset{(q-1)\times 1}{\mathbf{0} = \begin{vmatrix} 0 \\ \vdots \\ 0 \end{vmatrix}}; \tag{23}$$

This equation can also be written as follows

$$\mathbf{A}\mathbf{i} = -\mathbf{A}\Im \text{ or } \sum_{k=1}^{p} a_{jk} i_k = \sum_{k=1}^{p} a_{jk} \Im_k, \; j = 1...(q-1). \tag{24}$$

In this form of the Kirchhof's current law the current sources are specially emphasized.

3.1.5 Loop equations. Loop matrix

By applying the Kirchhof's voltage law one can write as many equations as many loops are in the circuit. However, some of these equations can be derived from others. Independence of loops equations or as it is said loop independence can be guaranteed if the every next loop is comprised of at least one new branch. The easiest way to do so is by using properties of graph tree. Graph tree is a set of edges which does not form any closed loops (cycles). One closed loop can be formed by adding one connection edge to the graph tree. This closed loop is comprised of this connection edge and some edges of graph tree.

For example, on Fig. 16. *a* loop *4* can be formed by connection edge *4* and tree edges *1* and *3*. On Fig. 16. *b* loop *5* can be formed by connection edge *5* and tree edges *1*, *2* and *3*. On Fig. 16. *c* loop *6* can be formed by connection edge - and tree edges *1* and *2*. Thus, *number of independent loops is determined by number of connection edges in the graph* $n = p - (q-1)$.

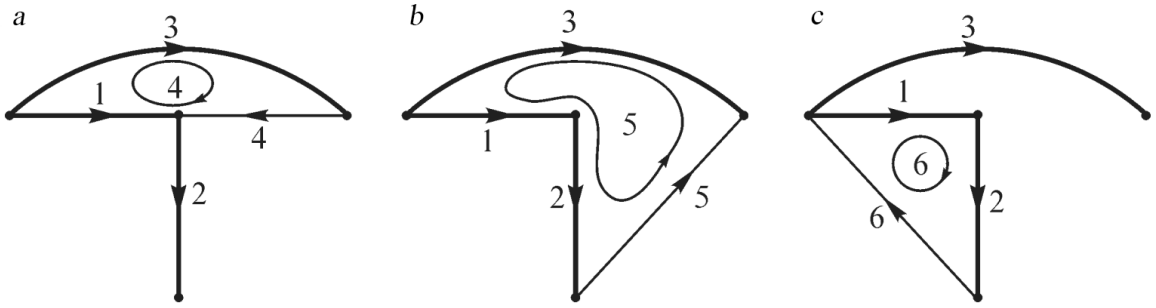

Fig. 16. Loop edges [16]

Let us write loop equations for diagram graph. Denote voltages of branches as $\tilde{u}_k$. Loop equation are numbered accordingly to connection edges. Direction of loop is the same with the direction of the connection edge. In the loop equations voltages have sign "plus" if loop and edge direction agrees and with "minus" in opposite case. Algebraically sign can be taken into account by coefficient c_{sk}. Coefficient $c_{sk} = 1$ if the *k*-th edge is included in the *s*-th loop with the same direction; $c_{sk} = -1$ if *k*-th edge is included in the *s*-th loop but with opposite direction; $c_{sk} = 0$ if *k*-th edge is not included in the *s*-th loop. By considering this Kirchhoff's voltage law can be written as follows

$$\sum_{k=1}^{p} c_{sk}\tilde{u}_k = 0, \;\; s = q...p \tag{25}$$

In view of the above, for the graph on Fig. 16. the system of three independent equations can be written

$$\begin{array}{lllll} \text{for loop 4} & -\tilde{u}_1 + \tilde{u}_3 + \tilde{u}_4 = 0, & c_{41} = -1, & c_{43} = 1, & c_{44} = 1; \\ \text{for loop 5} & \tilde{u}_1 + \tilde{u}_2 - \tilde{u}_3 = 0, & c_{51} = 1, & c_{52} = 1, & c_{53} = -1, \quad c_{55} = 1; \\ \text{for loop 6} & \tilde{u}_1 + \tilde{u}_2 + \tilde{u}_6 = 0, & c_{61} = 1, & c_{62} = 1, & c_{66} = 1. \end{array} \tag{26}$$

Let us form a table with c_{sk} coefficients. Rows of this table are numbered accordingly to connection edges and columns accordingly to graph edges. The rectangle matrix related to that table is called loop matrix. And coefficients are 0, 1 and -1 if the edge is correspondingly not included in the loop, is included and has the same direction as the loop or included and has opposite direction with the loop. Let us denote this matrix by $\mathbf{C}$.

Let us write voltages of graph edges as a matrix with one column and *p* rows:

$$\tilde{\mathbf{u}} = \left|\tilde{u}_k\right| = \begin{vmatrix} \tilde{u}_1 \\ \vdots \\ \tilde{u}_p \end{vmatrix}, \; k = 1 \ldots p. \tag{27}$$

Every row of loop matrix is a set of coefficients for voltages in a Kirchhoff's voltage law for loop which is formed by connection edge. Number of this connection edge determines the number of the row in the loop matrix. Therefore with respect to liner algebra rules every loop equation can be presented as follows

$$\underset{\text{vector } (1\times p)}{s\,\boxed{c_{s1}} \;\ldots\; \boxed{c_{sk}} \;\ldots\; \boxed{c_{sp}}} \times \begin{matrix} \boxed{\tilde{u}_1} \\ \vdots \\ \boxed{\tilde{u}_k} \\ \vdots \\ \boxed{\tilde{u}_p} \end{matrix} = \underset{\text{matrix } (1\times 1)}{c_{s1}\tilde{u}_1 + \ldots + c_{sk}\tilde{u}_k + \ldots + c_{sp}\tilde{u}_p} \quad \sum_{k=1}^{p} c_{sk}\tilde{u}_k = 0 \tag{28}$$

These matrix equations can be written for all *n* connection edges of the graph (rows of loop matrix). In matrix form system of these equations is written as follows

$$\mathbf{C}\tilde{\mathbf{u}} = \mathbf{0} \tag{29}$$

For circuit diagram graph (Fig. 12. *c*) we have

$$\mathbf{C}\tilde{\mathbf{u}} = \begin{array}{c} \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{|cccccc|} \hline -1 & & 1 & 1 & & \\ 1 & 1 & -1 & & 1 & \\ 1 & 1 & & & & 1 \\ \hline \end{array} \\ (3\times 6) \end{array} \times \begin{array}{|c|} \hline \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \\ \tilde{u}_5 \\ \tilde{u}_6 \\ \hline \end{array} \; (6\times 1) = \begin{array}{|c|} \hline -\tilde{u}_1 + \tilde{u}_3 + \tilde{u}_4 \\ \tilde{u}_1 + \tilde{u}_2 - \tilde{u}_3 + \tilde{u}_5 \\ \tilde{u}_1 + \tilde{u}_2 + \tilde{u}_6 \\ \hline \end{array} \; (3\times 1) = \begin{array}{|c|} \hline 0 \\ 0 \\ 0 \\ \hline \end{array} \; (3\times 1) = \mathbf{0} \tag{30}$$

Every row of matrix multiplication $\mathbf{C}\tilde{\mathbf{u}}$ determines the equation for the loop as stated by Kirchhoff's current law.

For *k*th generalized branch the following equation can be written

$$\tilde{u}_k = u_k + e_k, \tag{31}$$

where u_k - voltage on passive element of *k*th branch, and e_k-EMF in *k*th branch if they are directed correspondingly to graph edge.

Matrix form of system equations according to Kirchhoff's voltage law for voltages and EMF can be written as follows

$$\mathbf{C\tilde{u}} = \mathbf{Cu} + \mathbf{Ce} = \mathbf{0}, \tag{32}$$

where

$$\underset{(p\times 1)}{\tilde{\mathbf{u}}} = \begin{vmatrix} \tilde{u}_1 \\ \vdots \\ \tilde{u}_p \end{vmatrix}; \quad \underset{(p\times 1)}{\mathbf{u}} = \begin{vmatrix} u_1 \\ \vdots \\ u_p \end{vmatrix}; \quad \underset{(p\times 1)}{e} = \begin{vmatrix} e_1 \\ \vdots \\ e_p \end{vmatrix}; \quad \underset{n\times 1}{\mathbf{0}} = \begin{vmatrix} 0 \\ \vdots \\ 0 \end{vmatrix}; \tag{33}$$

This equation can also be written as follows

$$\mathbf{Cu} = \mathbf{Ce} \text{ or } \sum_{k=1}^{p} c_{sk} u_k = \sum_{k=1}^{p} c_{sk} e_k, \; s = q...p. \tag{34}$$

3.1.6 Current equations for circuit sections. Section matrix

The Kirchhoff's current law can be formulated not only for particular circuit nodes but also for set of nodes. For this case the surface for which the equation

$$\oint \mathbf{J} d\mathbf{s} = \sum i_k = 0 \tag{35}$$

is written will enclose the set of nodes and cut the circuit diagram into two parts. Let us depict sections on figures as closed dashed lines (for example Fig. 15. and Fig. 18.). There can be several sections in the circuit diagram. Each section is related to its equations which claims that the sum of currents cut by this section is equal to zero. As was stated earlier the number of independent equations by Kirchhoff's current law is equal to $q-1$. Therefore, the number of independent equations for sections is also equal to $q-1$ as each section equation can be get by summing corresponding node equations for nodes enclosed by section. To make the process of the sections selecting easier let us assume that the one section cut only one tree edge. Therefore, the number of section is equal to the number of tree edges. The sections can be numbered as corresponding tree edges are numbered. Let us also assume that term *edge direction* is a synonym to term *direction of the branch current.* The surface normal is directed inside or outside section depending on the direction of the tree edge. Therefore, tree edge current and edges currents are included in the equations for sections currents with plus sign if they are directed to the section the same

way as tree edge current. All other currents are included with minus sign. Currents which are not cut by the section boundary are not included in the equation. Algebraically sign can be taken into account by coefficient d_{mk}. Where m – number of tree edge, which is also number of the section; k – edge number. $d_{mk} = \pm 1$ if k-th edge is cut by m-th section boundary, and $d_{mk} = 0$ if k-th edge is not cut by m-th section boundary. By considering this equation for sections currents can be written as follows

$$\sum_{k=1}^{p} d_{mk}\tilde{i}_k = 0,\ m = 1...(q-1) \tag{36}$$

In view of the above, for the graph on Fig. 17. the system of three equations can be written

$$\begin{array}{lllll}
\text{for section 1 (Current out)} & \tilde{i}_1 + \tilde{i}_4 - \tilde{i}_5 - \tilde{i}_6 = 0, & d_{11} = 1, & d_{14} = 1, & d_{15} = -1, \quad d_{16} = -1; \\
\text{for section 2 (Current in)} & \tilde{i}_2 - \tilde{i}_5 - \tilde{i}_6 = 0, & d_{22} = 1, & d_{25} = -1, & d_{26} = -1. \\
\text{for section 3 (Current in)} & \tilde{i}_3 - \tilde{i}_4 + \tilde{i}_5 = 0, & d_{33} = 1, & d_{34} = -1, & d_{35} = 1.
\end{array} \tag{37}$$

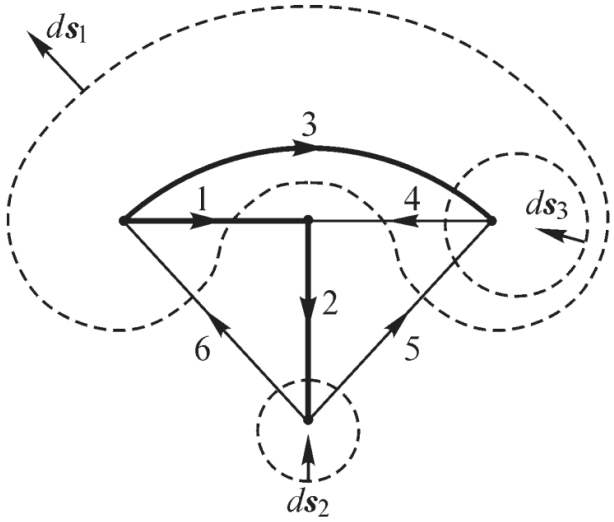

Fig. 17. Circuit and its sections [16]

Let us form a table with d_{mk} coefficients. Rows of this table are numbered accordingly to tree edges numbers and columns accordingly to graph edges. The rectangle matrix related to that table is called sections matrix. Section cut only one tree edge and any number of connection edges. Table coefficients are:

0 – if the section does not cut the edge

1 – if the section cut edge and edge and surface normal direction are similar

-1 – if the section cut edge and edge and surface normal direction are opposite

Let us denote this matrix by $\mathbf{D}$.Let us write currents of graph edges as a matrix with one column and *p* rows:

$$\tilde{\mathbf{i}} = \left|\tilde{i}_k\right| = \begin{vmatrix} \tilde{i}_1 \\ \vdots \\ \tilde{i}_p \end{vmatrix}, \; k = 1...p. \tag{38}$$

Every row of sections matrix is a set of coefficients for currents in a system of sections equations, number of which is the same with the number of tree edge. Therefore, with respect to linear algebra rules every nodal equation can be presented as follows

$$\underset{\text{vector } (1\times p)}{m\boxed{d_{m1}} \; ... \; \boxed{d_{mk}} \; ... \; \boxed{d_{mp}}} \times \begin{matrix} \boxed{\tilde{i}_1} \\ \vdots \\ \boxed{\tilde{i}_k} \\ \vdots \\ \boxed{\tilde{i}_p} \end{matrix} = \underset{\text{matrix } (1\times 1)}{d_{m1}\tilde{i}_1 + ... + d_{mk}\tilde{i}_k + ... + d_{mp}\tilde{i}_p} \quad \sum_{k=1}^{p} d_{mk}\tilde{i}_k = 0 \tag{39}$$

These matrix equations can be written for all $q-1$ edges of the graph. In matrix form system of these equations is written as follows

$$\mathbf{D}\tilde{\mathbf{i}} = \mathbf{0} \tag{40}$$

For circuit diagram graph (Fig. 17.) we have

$$\mathbf{D}\tilde{\mathbf{i}} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{|cccccc|} \hline 1 & & & 1 & -1 & -1 \\ & 1 & & & -1 & -1 \\ & & 1 & -1 & 1 & \\ \hline \end{array} \\ (3\times 6) \end{array} \times \begin{array}{|c|} \hline \tilde{i}_1 \\ \tilde{i}_2 \\ \tilde{i}_3 \\ \tilde{i}_4 \\ \tilde{i}_5 \\ \tilde{i}_6 \\ \hline \end{array} \; (6\times 1) = \begin{array}{|c|} \hline \tilde{i}_1 + \tilde{i}_4 - \tilde{i}_5 - \tilde{i}_6 \\ \tilde{i}_2 - \tilde{i}_5 - \tilde{i}_6 \\ \tilde{i}_3 - \tilde{i}_4 + \tilde{i}_5 \\ \hline \end{array} \; (3\times 1) = \begin{array}{|c|} \hline 0 \\ 0 \\ 0 \\ \hline \end{array} \; (3\times 1) = \mathbf{0} \tag{41}$$

Every row of matrix multiplication $\mathbf{D}\tilde{\mathbf{i}}$ determines the equation for sections currents as stated by Kirchhoff's current law.

For *k*th generalized branch the following equation can be written

$$\tilde{i}_k = i_k + \Im_k, \tag{42}$$

where i_k - current in passive elements of *k*-th branch, and $\Im_k$ - current of current source in *k*-th branch if they are directed from the same node.

Matrix form of system equations for sections currents can be written as follows

$$\mathbf{D}\tilde{\mathbf{i}} = \mathbf{D}\mathbf{i} + \mathbf{D}\Im = \mathbf{0}, \tag{43}$$

where

$$\underset{(p\times 1)}{\tilde{\mathbf{i}}} = \begin{vmatrix} \tilde{i}_1 \\ \vdots \\ \tilde{i}_p \end{vmatrix}; \quad \underset{(p\times 1)}{i} = \begin{vmatrix} i_1 \\ \vdots \\ i_p \end{vmatrix}; \quad \underset{(p\times 1)}{\Im} = \begin{vmatrix} \Im_1 \\ \vdots \\ \Im_p \end{vmatrix}; \quad \underset{(q-1)\times 1}{\mathbf{0}} = \begin{vmatrix} 0 \\ \vdots \\ 0 \end{vmatrix}; \tag{44}$$

This equation can also be written as follows

$$\mathbf{D}\mathbf{i} = -\mathbf{D}\Im. \tag{45}$$

In this form current sources are specially emphasized.

3.1.7 Interconnection among nodal connections, loops and sections matrices

All three matrices described earlier can be transformed to each another. However to make it possible graph edges have to be numbered in a certain order. Let us further number tree edges from 1 to $(q-1)$ and graph connection edges from *q* to *p*. By considering this matrices **A**, **C** and **D** can be presented as two submatrices (or two blocks):

$$\mathbf{A} = \begin{array}{c} \\ 1 \\ \vdots \\ q-1 \end{array} \begin{array}{c} \begin{array}{cc} 1 \ldots (q-1) & q \ldots p \end{array} \\ \left[\begin{array}{c|c} \mathbf{A}_1 & \mathbf{A}_2 \end{array}\right] \\ \begin{array}{cc} (q-1)\times(q-1) & (q-1)\times n \end{array} \end{array}; \quad \mathbf{C} = \begin{array}{c} \\ q \\ \vdots \\ p \end{array} \begin{array}{c} \begin{array}{cc} 1 \ldots (q-1) & q \ldots p \end{array} \\ \left[\begin{array}{c|c} \mathbf{C}_1 & \mathbf{C}_2 \end{array}\right] \\ \begin{array}{cc} n\times(q-1) & (n\times n) \end{array} \end{array}; \tag{46)(47}$$

$$\mathbf{D} = \begin{array}{c} \\ 1 \\ \vdots \\ q-1 \end{array} \begin{array}{c} \begin{array}{cc} 1 \ldots (q-1) & q \ldots p \end{array} \\ \left[\begin{array}{c|c} \mathbf{D}_1 & \mathbf{D}_2 \end{array}\right] \\ \begin{array}{cc} (q-1)\times(q-1) & (q-1)\times n \end{array} \end{array} \tag{48}$$

Currents (voltages) can also be presented as two column blocks – two submatrices. The first block contain currents (voltages) of tree edges with numbers from 1 to $q-1$, the second block – with numbers from *q* to *p*

$$\tilde{\mathbf{i}} = \begin{matrix} 1 \\ \vdots \\ (q-1) \\ q \\ \vdots \\ p \end{matrix} \left[\begin{array}{c} \tilde{\mathbf{i}}_1 \\ \hline \tilde{\mathbf{i}}_2 \end{array}\right] \begin{matrix} (q-1)\times 1 \\ \\ (n\times 1) \end{matrix} ; \quad \tilde{\mathbf{u}} = \begin{matrix} 1 \\ \vdots \\ (q-1) \\ q \\ \vdots \\ p \end{matrix} \left[\begin{array}{c} \tilde{\mathbf{u}}_1 \\ \hline \tilde{\mathbf{u}}_2 \end{array}\right] \begin{matrix} (q-1)\times 1 \\ \\ (n\times 1) \end{matrix} \qquad (49)(50)$$

By this kind of splitting matrix equations can be written as a product of these submatrices. In according with Kirchhoff's current law

$$\mathbf{A}\tilde{\mathbf{i}} = \left[\begin{array}{c|c} \mathbf{A}_1 & \mathbf{A}_2 \end{array}\right] \times \left[\begin{array}{c} \tilde{\mathbf{i}}_1 \\ \hline \tilde{\mathbf{i}}_2 \end{array}\right] = \mathbf{A}_1\tilde{\mathbf{i}}_1 + \mathbf{A}_2\tilde{\mathbf{i}}_2 = \mathbf{0} \qquad (51)$$

Where:

$\mathbf{A}_1$ - square submatrix with dimensions $(q-1)\times(q-1)$, which has $(q-1)$ columns and $(q-1)$ rows related to $(q-1)$ tree edges and $(q-1)$ nodes correspondingly.

$\tilde{\mathbf{i}}_1$ - submatrix-column with $(q-1)$ rows related to $(q-1)$ tree edges.

Equality of submatrix $\mathbf{A}_1$ columns and submatrix-column $\tilde{\mathbf{i}}_1$ rows allows us to write matrix product $\mathbf{A}_1\tilde{\mathbf{i}}_1$. The same is right for matrix product $\mathbf{A}_2\tilde{\mathbf{i}}_2$ because $\mathbf{A}_2$ has *n* columns and $\tilde{\mathbf{i}}_2$ has *n* rows which are related to number of connection edges in graph.

Similarly this splitting can be performed for **C** and **D** matrices.

$$\mathbf{C}\tilde{\mathbf{u}} = \left[\begin{array}{c|c} \mathbf{C}_1 & \mathbf{C}_2 \end{array}\right] \times \left[\begin{array}{c} \tilde{\mathbf{u}}_1 \\ \hline \tilde{\mathbf{u}}_2 \end{array}\right] = \mathbf{C}_1\tilde{\mathbf{u}}_1 + \mathbf{C}_2\tilde{\mathbf{u}}_2 = \mathbf{0} \qquad (52)$$

$$\mathbf{D}\tilde{\mathbf{i}} = \left[\begin{array}{c|c} \mathbf{D}_1 & \mathbf{D}_2 \end{array}\right] \times \left[\begin{array}{c} \tilde{\mathbf{i}}_1 \\ \hline \tilde{\mathbf{i}}_2 \end{array}\right] = \mathbf{D}_1\tilde{\mathbf{i}}_1 + \mathbf{D}_2\tilde{\mathbf{i}}_2 = \mathbf{0} \qquad (53)$$

where:

$\tilde{\mathbf{i}}_1$ and $\tilde{\mathbf{u}}_1$ - column submatrices of the tree edges currents and voltages,

$\tilde{\mathbf{i}}_2$ and $\tilde{\mathbf{u}}_2$ - column submatrices of the connection edges currents and voltages.

In loop matrix $\mathbf{C}$ row number is determined by the number of the connection edge. Taking into account that there is only one connection edge in the loop therefore it is obvious that submatrix $\mathbf{C}_2$ has only one nonzero element in the row. This element is placed in the column determined by the number of the connection edge which form this loop. Considering above all nonzero elements of the submatrix $\mathbf{C}_2$ equal to one will be placed on the main diagonal of $\mathbf{C}_2$ submatrix. This kind of matrix is called *identity* matrix. Les us denote identity matrix as **1**. $\mathbf{C}_2 = \mathbf{1}$ submatrix has dimensions $(n \times n)$. Conventionally identity matrix is denoted by **I**. However, in order to avoid confusing with current we will use symbol **1.**

In section matrix rows are numbered accordingly to numbers of tree edges. Taking into account that there is only one tree edge in the section, every submatrix $\mathbf{D}_1$ row will have only one nonzero element. These elements will be placed on the main diagonal of $\mathbf{D}_1$ submatrix. Therefore submatrix $\mathbf{D}_1$ is a Identity matrix with dimensions $(q-1)\times(q-1)$.

Considering above **C** and **D** matrices can be written as follows

$$\mathbf{C} = \begin{array}{c} \\ q \\ \vdots \\ p \end{array} \begin{array}{c} \begin{array}{cc} 1 \ldots (q-1) & q \ldots p \end{array} \\ \left[\begin{array}{c|c} \mathbf{C}_1 & \mathbf{1} \end{array}\right] \\ \begin{array}{cc} n\times(q-1) & (n\times n) \end{array} \end{array} ; \quad \mathbf{D} = \begin{array}{c} \\ 1 \\ \vdots \\ q-1 \end{array} \begin{array}{c} \begin{array}{cc} 1 \ldots (q-1) & q \ldots p \end{array} \\ \left[\begin{array}{c|c} \mathbf{1} & \mathbf{D}_2 \end{array}\right] \\ \begin{array}{cc} (q-1)\times(q-1) & (q-1)\times n \end{array} \end{array} \qquad (54)(55)$$

Every section cuts only one tree edge and some graph connection edges (Fig. 18.). Tree edge (*m* on the Fig. 18.) which determines the section is a part of loops (*j* and *s)* which are formed by those connection edges which are cut by the section because cutting this tree edge will cut all loops formed by connection edges.

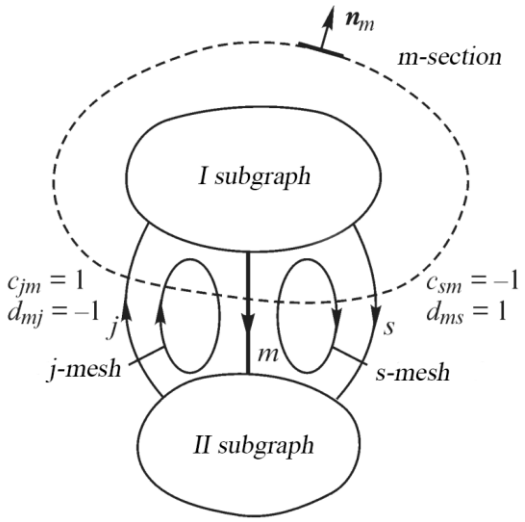

Fig. 18. For demonstration of interconnection principle [16]

Two subgraphs on Fig. 18. are formed by cutting *m* edge. Edge *m* is included in loops formed by *s* and *j* connection edges with signs $c_{sm} = -1$ and $c_{jm} = 1$ correspondingly. Connections edges *s* and *j* are included in *m*-th section with signs $d_{ms} = 1$ and $d_{mj} = -1$ correspondingly. Therefore, for all loops and sections it can be noticed that columns of submatrix $\mathbf{C}_1$ can be formed by rows of the submatrix $\mathbf{D}_2$ if all nonzero elements are with opposite signs. Considering above we can write

$$\mathbf{C}_1 = -\mathbf{D}_2^{\mathrm{T}} \text{ or } \mathbf{D}_2 = -\mathbf{C}_1^{\mathrm{T}} \tag{56}$$

Let us make a substitution $\mathbf{C}_1 = \mathbf{F}$. Now loop and section matrices can be written as follows

$$\mathbf{C} = \left[\begin{array}{c|c} \mathbf{F} & \mathbf{1} \end{array}\right]; \ \mathbf{D} = \left[\begin{array}{c|c} \mathbf{1} & -\mathbf{F}^{\mathrm{T}} \end{array}\right] \tag{57)(58}$$

Therefore, to get matrices **C** and **D** it is enough just to form one submatrix **F**. For example for graph on the Fig. 19. we have

$$\mathbf{D} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \left[\begin{array}{cccc:cccccc} 1 & & & & 1 & -1 & -1 & & & \\ & 1 & & & -1 & 1 & 1 & & 1 & 1 \\ & & 1 & & & 1 & & -1 & & 1 \\ & & & 1 & 1 & -1 & & 1 & -1 & -1 \end{array}\right] \end{array} = \left[\begin{array}{c|c} \mathbf{1} & -\mathbf{F}^{\mathrm{T}} \end{array}\right] \tag{59}$$

Therefore,

$$\mathbf{F} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 5 & -1 & 1 & & -1 \\ 6 & 1 & -1 & -1 & 1 \\ 7 & 1 & -1 & & \\ 8 & & & 1 & -1 \\ 9 & & -1 & & 1 \\ 10 & & -1 & -1 & 1 \end{array} \quad \text{and} \quad \mathbf{C} = \begin{array}{c|cc} & 1\ldots4, & 5\ldots10 \\ 5 & & \\ \vdots & \mathbf{F} & \mathbf{1} \\ 10 & & \end{array}. \qquad (60)(61)$$

From Kirchhoff's laws we have $\mathbf{A_1}\tilde{\mathbf{i}}_1 = -\mathbf{A_2}\tilde{\mathbf{i}}_2$ and $\mathbf{D_1}\tilde{\mathbf{i}}_1 = -\mathbf{D_2}\tilde{\mathbf{i}}_2$. However $\mathbf{D}_1 = \mathbf{1}$, therefore $\tilde{\mathbf{i}}_1 = -\mathbf{D_2}\tilde{\mathbf{i}}_2$.

Submatrix $\mathbf{A}_1$ is a square matrix. Therefore, it is possible to find such matrix $\mathbf{A}_1^{-1}$ that $\mathbf{A}_1^{-1} \cdot \mathbf{A}_1 = \mathbf{1}$. Matrix $\mathbf{A}_1^{-1}$ is called inverse. Let us multiply first matrix equation and $\mathbf{A}_1^{-1}$ on the left

$$\mathbf{A_1}\tilde{\mathbf{i}}_1 = -\mathbf{A_2}\tilde{\mathbf{i}}_2 \Rightarrow \mathbf{A_1^{-1}A_1}\tilde{\mathbf{i}}_1 = -\mathbf{A_1^{-1}A_2}\tilde{\mathbf{i}}_2 \Rightarrow \mathbf{1}\tilde{\mathbf{i}} = -\mathbf{A_1^{-1}A_2}\tilde{\mathbf{i}}_2 = -\mathbf{D_2}\tilde{\mathbf{i}}_2 \Rightarrow$$
$$\Rightarrow \mathbf{D_2} = \mathbf{A_1^{-1}A_2} \qquad (62)$$

And also

$$-\mathbf{F}^{\mathrm{T}} = \mathbf{A_1^{-1}A_2}. \qquad (63)$$

Last two equations are very important because they allow us to transform connections matrix **A** to loop matrix **C** and sections matrix **D** by performing matrix manipulations. It is important because computer requires formalized procedure of forming system of equations. Submatrix $\mathbf{D}_2$ always exists for graph, therefore taking into account equation $\mathbf{D_2} = \mathbf{A_1^{-1}A_2}$ and existence of $\mathbf{A}_2$ matrix, it can be noted that inverse matrix $\mathbf{A}_1^{-1}$ also always exists.

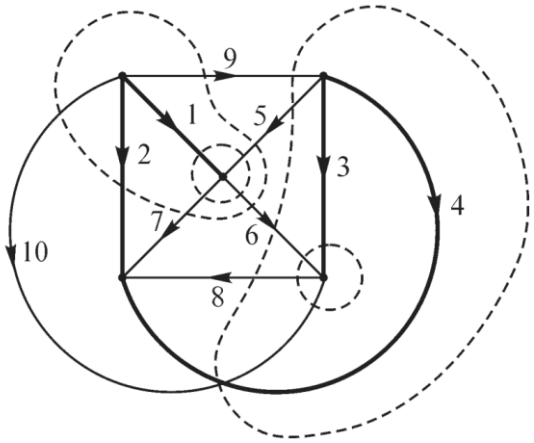

Fig. 19. Circuit graph and its sections [16]

3.2 Implementation block

This section describes algorithm of circuit analysis procedure and its implementation in MatLab. MatLab, which stands for Matrix Laboratory, is a software developed especially to operate with matrices, thus using it benefits in a very high computational performance towards this problem.

3.2.1 Algorithm

As was mentioned in 2.1, there are two classical methods for analyzing circuits based on two Kirchhoff's laws. The answer on question: «what method is more suitable for problem?» depends on properties of circuit in study. Saying «more suitable» only computational performance is considered. Two methods require different number of equations to be solved by computer, which is $(q-1)$ for Nodal Analysis and $p-(q-1)$ for Mesh Analysis. Therefore, at first these two values have to be compared and the smallest value determines method which will require less equations for problem solving.

3.2.2 Choose method. Graphs

Topological structure of circuit is fully determined by such entity as graph. There are several computer representation of graphs. First of them is incidence matrix. Incidence matrix is a matrix that shows the relationship between two classes of objects. The first class is nodes and the second is edges, the matrix has one row for each element of nodes set and one column for each element of edges set. The entry in row *i* and column *j* is 1 if *i* node and *j* edge are related (called incident in this context) and 0 if they are not [20]. Building this matrix for graph and eliminating one row will result in Matrix of nodal connections, which can be used for circuit analysis, in other words, for nodal analysis or being transformed to mesh matrix - for mesh analysis. Therefore, the first task is to set incidence matrix of graph. This task can be performed element-wise by iteratively setting elements of matrix to 1 or 0. However, this approach is not effective because it requires one to set $(p \cdot q)$ elements of matrix. Despite the necessity of having matrix of nodal connections (reduced incidence matrix) for future computation, there is no need to set elements of this table element-wise. There is second approach to store information about matrix in computer, which is called table of edges (or element-wise input method). In this approach there should be set only two numbers for each edge, which are the numbers of

nodes. Therefore, using this approach it is much easier to create a graph because it requires setting only $(2 \cdot p)$ values in computer. Moreover, exactly this approach is used as default in MatLab to create a graph. It should be noticed that it is not the only one way to create a graph in MatLab, but the simplest for graph with many nodes and edges. One more advantage of this approach is that it operates with such an entity as edge, which represent a branch of electrical circuit and has its own resistance, EMF and current what is more straightforward for people than matrix of nodal connections, which is actually a big table of ones and zeros.

After branch table has been set it can be easily converted to incidence matrix and further to matrix of nodal connections by elimination of any one row. At this stage nodal analysis already can be performed. However, mesh analysis requires mesh matrix, which can be get by some matrix manipulations which was described in previous section in details. In order to make these manipulations work columns (edges) of matrix of nodal connections have to be reordered in a certain way. First $(q-1)$ columns are related to $(q-1)$ edges of graph tree and last $(p-q)$ columns to graph connections. MatLab has built-in function `minspantree`, which extract a tree (as a graph) from the main graph [21]. In allows us easily number all edges of tree and then number remaining edges, which are graph connections. After this reordering matrix of nodal connections **A** can be split into two submatrices $\mathbf{A}_1$ and $\mathbf{A}_2$:

```
sz = min(size(A)); % Number of A1 rows and columns

A1 = A(1:sz, 1:sz );
A2 = A(1:sz, sz+1:end );
```

In order to get $\mathbf{C} = [\mathbf{C}_1 \quad \mathbf{C}_2] = [\mathbf{C}_1 \quad \mathbf{1}] = [-\mathbf{F}^t \quad \mathbf{1}]$, $\mathbf{F}$ has to be derived as follows:

```
F = inv(A1)*A2;
```

And then:

```
C1 = -F';
C2 = speye(length(A)-sz); % Identity  matrix
C = [C1, C2];
```

Now mesh analysis also can be performed.

3.2.3 Sparse matrices

Despite edges table approach, which was chosen just to set information about circuit to computer, this edges table will still be converted to incidence matrix and then all computation also will require matrix manipulation. It means that computer has to store matrices in its memory and perform computation with these matrices. The size of memory required for storing grows with the size of these matrices. Standard data type in MatLab is *double*, which requires 8 bytes of memory to be allocated for one element of this type. Therefore, size of matrix

$$memory = 8 \cdot (m \cdot n),\ bytes \tag{64}$$

For example, for matrix of size $(10,000 \times 50,000)$, which is a matrix for circuit with 10,000 nodes and 50,000 branches MatLab will allocate 4 Gb of memory. Moreover, even if matrix will contain less elements circuit analysis methods require to store several matrix in memory per time, for example, for multiplication. All of that lead to huge performance decrease or even to inability of MatLab to allocate memory what produces error.

Let us assume two matrices **M1** and **M2**

$$\mathbf{M1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{M}2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Multiplication of these two matrices requires multiplication every row of first matrix on every column of second matrix as follows

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} (1\cdot1+0\cdot1+0\cdot1) & (1\cdot1+0\cdot1+0\cdot1) & (1\cdot1+0\cdot1+0\cdot1) \\ (0\cdot1+1\cdot1+0\cdot1) & (0\cdot1+1\cdot1+0\cdot1) & (0\cdot1+1\cdot1+0\cdot1) \\ (0\cdot1+0\cdot1+1\cdot1) & (0\cdot1+0\cdot1+1\cdot1) & (0\cdot1+0\cdot1+1\cdot1) \end{bmatrix}$$

Here were performed 27 multiplications and 18 additions what in total equals to 45 operations. However, many of these operations are not necessary, because, as can be noticed operations with zeros give no changes in the result matrix. If these operations could be eliminated it would result into only 9 operations in total what is 5 times faster. This number will only increase for bigger matrices. Some matrix manipulations, for

example, solving system of linear equations also make these unnecessary operations on zeros and their number can be very big.

Fortunately, all issues mentioned above can be overcame by using another data type, which is called *sparse matrix*. In mathematics sparse matrix is a matrix, which has many zero elements. Sparsity of a matrix can be characterized as follows

$$Sparsity = 1 - nnz / n \tag{65}$$

Where,

nnz – number of nonzero elements

n – number of all matrix elements

There is no exact threshold value of *Sparsity* after which matrix can be called sparse, because matrix can be interpreted as sparse by some different reasons. In our case, these reasons are memory required for matrix storing and speed of computation.

There is no mathematical difference between sparse matrix data type and ordinary matrix. The only difference between them is that sparse matrix data type does not store zero elements. In addition, some functions towards this data type can be overloaded to perform operations only for nonzero elements. Therefore, a lot of memory can be saved and performance can be gained.

In MatLab, sparse matrix is comprised of following parts [22, 23]:

1. Vector of nonzero elements of type double (8 bytes);
2. Vector of row indices for first vector. Indices have integer type (8 bytes);
3. Vector of row indices for first elements in every column. Indices have integer type (8 bytes);
4. Scalar – the number of all nonzero elements in matrix. Integer type (8 bytes)

Summing up all parts give the formula for calculating the size of the sparse matrix as follows

$$memory = (8+8) \cdot nnz + 8 \cdot columns + 8, \; bytes \tag{66}$$

For example, circuit with 10’000 nodes and 50.000 branches requires $q \times b = 10,000 \times 50,000$ incidence matrix in which every column contain only two elements. Therefore, using ordinary matrix we will store $nnz = 2 \cdot 50,000 = 100,000$ nonzero elements and $10,000 \cdot 40,000 - 100,000 = 399,990,000$ zero elements. Sparsity of

this matrix is $Sparsity = 1 - \frac{100,000}{399,900,000} = 0.999,974,9$ and it will require 4 Gb of memory. This matrix can be also stored in sparse matrix data type, which will require $memory = (8+8) \cdot nnz + 8 \cdot columns + 8 = (8+8) \cdot 100,000 + 8 \cdot 50,000 + 8 = 2,000,008$ what is only 2 Mb.

All reasons mentioned above make use of sparse matrix data type crucial in circuit analysis with nodal or mesh methods.

3.2.4 Nodal and Mesh analysis

Nodal or mesh analysis operates with such entity as generalized branch. Generalized branch is comprised of two parallel branches: one with resistor and EMF source in series and another with current source (Fig. 20.) [16, 18].

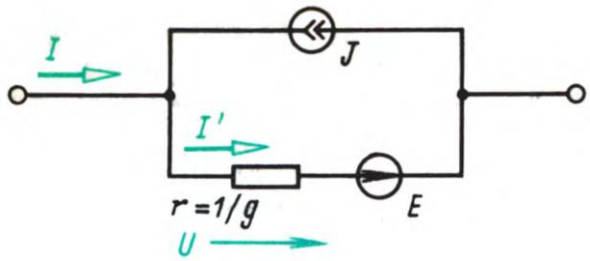

Fig. 20. Generalized branch [3]

Every circuit branch can be presented as generalized by assuming ***J*** zero if branch does not have current source in parallel or **E** equals to zero if branch has no EMF source.

To set information about circuit to computer it has to be presented as set of generalized branches. Each branch can be described by following parameters:

1. *s* – Number of staring node
2. *t* – Number of target node
3. *R* – Resistance of a branch
4. *E* – EMF source value
5. *J* – Current source value

By collecting these parameters for all generalized branches of the circuit they can be presented as vectors **s**, **t**, **R**, **E**, **J**. These five vectors are enough to describe the circuit. After having been set to computer **s** and **t** vectors can easily be converted to incidence matrix of the related graph and next to nodal connections matrix by elimination of one row. Then classic nodal method can be applied. The algorithm can be presented as follows

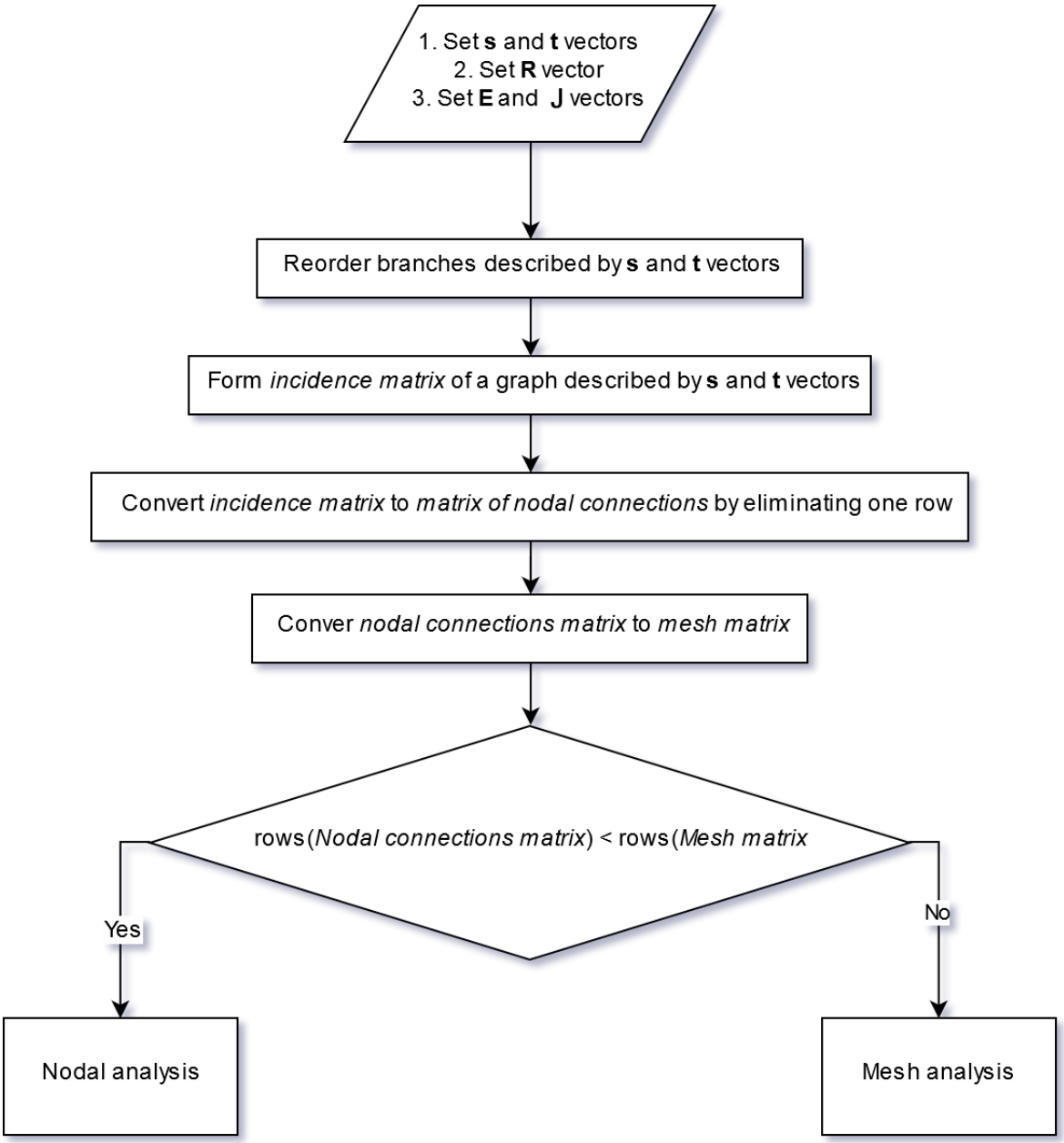

Fig. 21. Circuit analysis algorithm

4 INPUT DATA SPECIFICATION AND METHOD VALIDATION

As can be seen from the Fig. 21. The very first step requires setting information about 1. Circuit topology, 2. Reluctance elements and 3. Sources to computer. Approaches which were used for each of the input are discussed in this chapter.

4.1 Circuit topology

Circuit topology is fully described by its graph. Graph of the circuit contain all the information about circuit branches and nodes. Therefore, to set information about circuit topology a related graph has to be set to computer. Despite of different ways to store graph in the computer memory the most efficient one for setting it to computer is by s and t vectors. By «efficient» the time for manual setting is implied. After having been drawn on a paper circuit nodes have to be assigned with numbers. Then s and t vectors can be filled. Index of element in these vectors denotes their order number. For example for the circuit on Fig. 22. *a*, **s** and **t** vectors

$$\mathbf{s} = [5 \quad 2 \quad 3 \quad 4 \quad 2 \quad 3 \quad 6 \quad 6 \quad 6] \quad - (\text{Starting})$$

$$\mathbf{t} = [1 \quad 1 \quad 1 \quad 3 \quad 4 \quad 5 \quad 4 \quad 5 \quad 2] \quad - (\text{Target})$$

$$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad - \text{ branches indices}$$

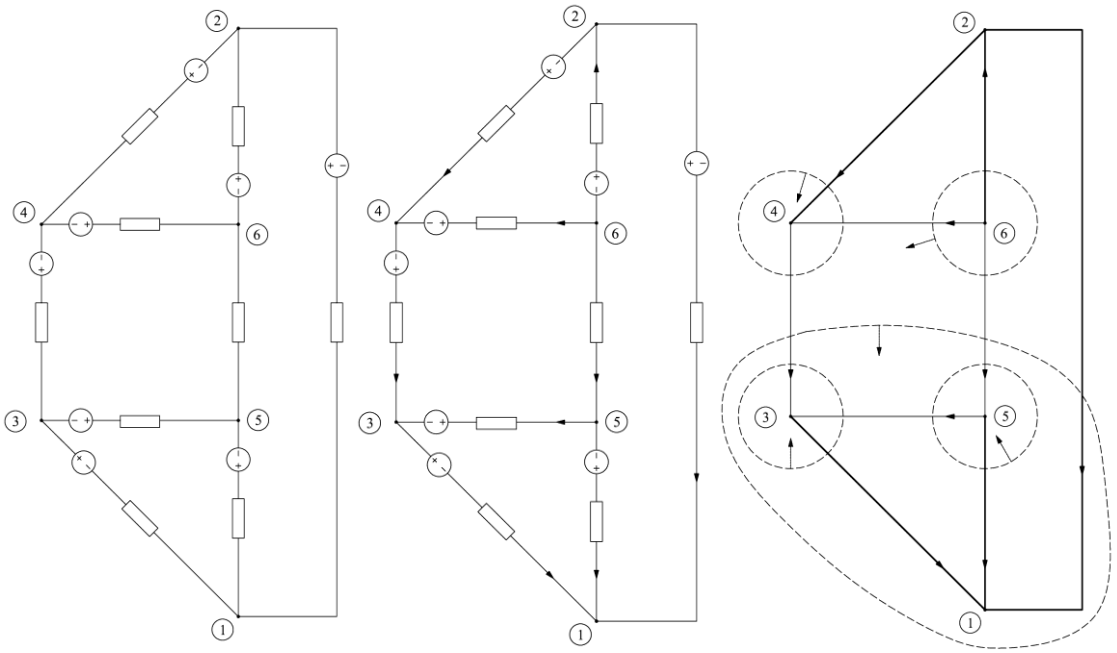

Fig. 22. a) Circuit, b) Circuit with directed branches, c) graph of the circuit

Moreover, indices of **s** and **t** vectors determine the direction of the branches (Fig. 22. *b* and *c*) . For example, for the third branch index is 3 and its nodes are 3 and 1. Therefore, if current value got from the CircuitAnalysis procedure is -1.58 that means that current flows from node 1 to node 3 with magnitude of 1.58.

4.2 Input values of resistances (reluctances)

Each branch described in s and t vectors has its resistance. Therefore, the vector with the same number of values as **s** and **t** has to be created. For circuit on Fig. 22. It is easily done by hand. For example, vector **R** can be presented as follows

$$\begin{aligned}
\mathbf{s} &= [\ 5\ \ 2\ \ 3\ \ 4\ \ 2\ \ 3\ \ 6\ \ 6\ \ 6\ ] \\
\mathbf{t} &= [\ 1\ \ 1\ \ 1\ \ 3\ \ 4\ \ 5\ \ 4\ \ 5\ \ 2\ ] \\
\mathbf{R} &= [\ 33\ \ 37\ \ 6\ \ 37\ \ 26\ \ 4\ \ 12\ \ 22\ \ 39\ ]
\end{aligned}$$

4.3 Input values of current (flux) sources and electromotive (magnetomotive) forces

Each branch additionally to its resistance has to have current source and electromotive force (chapter 3.2.4). However, depending on the particular circuit values of these elements can be zero, which means its absence. Ideal current source has inner resistance equal to infinite, thus setting the value of this source to zero is equivalent to replacing current source by break in a circuit. The same way ideal electromotive force elements can be replaced by a short circuit, because inner resistance of the source is zero.

For circuit without current sources as on Fig. 22. Vectors **E** and **J** can be also set manually as follows

$$\begin{aligned}
\mathbf{s} &= [\ 5\ \ 2\ \ 3\ \ 4\ \ 2\ \ 3\ \ 6\ \ 6\ \ 6\ ] \\
\mathbf{t} &= [\ 1\ \ 1\ \ 1\ \ 3\ \ 4\ \ 5\ \ 4\ \ 5\ \ 2\ ] \\
\mathbf{R} &= [\ 33\ \ 37\ \ 6\ \ 37\ \ 26\ \ 4\ \ 12\ \ 22\ \ 39\ ] \\
\mathbf{E} &= [\ 19\ \ 28\ \ -19\ \ 38\ \ 24\ \ 20\ \ -27\ \ -11\ \ 16\ ] \\
\mathbf{J} &= [\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ]
\end{aligned}$$

Now the full set of input vectors is prepared for CircuitAnalysis procedure.

4.4 Circuit analysis

After input vectors having been filled CircuitAnalysis procedure can be run. The procedure has syntax as follow

```
[Nodes, Branches] = CircuitAnalysis( s, t, R, E, J);
```

It has two output variables, which are Nodes and Branches. These variable of MatLab table type and contain detail information about circuit. For circuit on Fig. 22. Nodes and Branches tables are presented on Fig. 23. and Fig. 24. accordingly.

```
Nodes =
  6×2 table
    Num    Potential
    ___    _________
     1      -13.403
     2      -27.927
     3      -3.8987
     4      -23.743
     5       7.8078
     6            0
```

Fig. 23. Nodes table as output from CircuitAnalysis procedure

```
Branches =
  9×8 table
    Branch    s    t    R     E     J      U         I
    ______    _    _    __    ___    _    _______    _______
      1       5    1    33     19    0     21.21      1.2185
      2       2    1    37     28    0    -14.525     0.3642
      3       3    1     6    -19    0     9.5038    -1.5827
      4       4    3    37     38    0    -19.844     0.4907
      5       2    4    26     24    0    -4.1844     0.76214
      6       3    5     4     20    0    -11.706     2.0734
      7       6    4    12    -27    0     23.743    -0.27144
      8       6    5    22    -11    0    -7.8078    -0.8549
      9       6    2    39     16    0     27.927     1.1263
```

Fig. 24. Branches table as output from CircuitAnalysis procedure

As can be seen algorithm described in previous chapter was implemented inside this procedure and it give correct values of nodes potentials and branches currents.

CircuitAnalysis procedure always display in the Command Window the method

which was used for particular circuit. Moreover, CircuitAnalysis procedure is able to interpret next Name-Value pair of input arguments:

1. Method of solving.

   Purpose: allow to manually set method to be applied

   Name: 'Method'

   Values: 'Auto' (Default) | 'Nodal' | 'Mesh'

   Syntax: `CircuitAnalysis( s, t, R, E, J, 'Method', 'Auto')`

2. Condition number

   Purpose: disable computation of condition number because it can take much time

   Name: 'ConditionNumber'

   Values: 'on' (default) | 'off'

   Syntax: `CircuitAnalysis( s, t, R, E, J, 'ConditionNumber', 'on')`

3. Order of branches in Branches table (applies only for mesh method)

   Purpose: to sort branch order

   Name: 'Order'

   Values: 'Branches' (Default) | 'Nodes' | 'mst' |

   'Branches' - order as was set in s and t vectors

   'Nodes' - ascending order of nodes (sorted by algorithm built in a 'graph' function)

   'mst' - order as was sorted by convention for forming Mesh matrix

   Syntax: `CircuitAnalysis( s, t, R, E, J, 'Order', 'Branches')`

4. Symbolic matrix of coefficients

   Purpose: to display matrix of coefficients in a symbolic way

   Name: 'Coefficients'

   Values: 'Hide' (Default) | 'Show'

   Syntax: `CircuitAnalysis( s, t, R, E, J, 'Coefficients', 'Hide')`

All these Name-Value pairs can be used together and in any order.

4.5 Considerations for big circuits and validation of the method

Let us assume a structure to create a reluctance network for it. Fig. 25. Shows the periodic linear structure and its sizes in mm. Active length is 1 m. Magnetic remanence of magnets is 1.2 T. Relative permeability of steel is assumed constant and equals to 7500. Relative air and magnets permeability is 1.

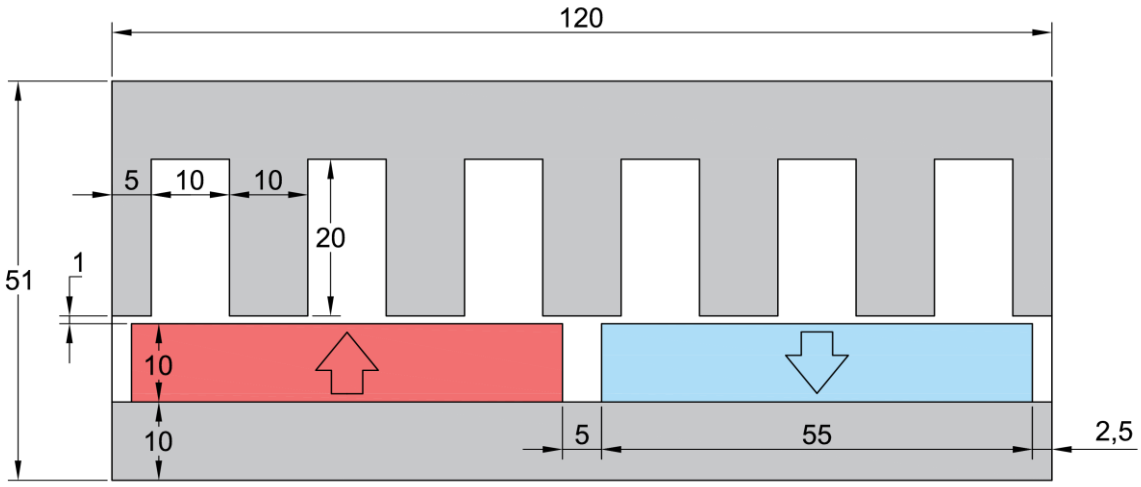

Fig. 25. Linear structure

Gray color on the structure depicts steel, white – air, red and blue – magnets. The aim is to get radial component of the flux density in the air-gap region (above magnets and below teeth). As was mentioned in Chapter one to simplify the reluctance network let us assume all reluctance elements the same size. Then a mesh-based reluctance network can be created to cover the structure. It has to be noticed that the size of every element is determined by the size of the smallest element. The smallest elements have to be fitted in the air-gap region, thus other regions of the structure will be also filled with elements of air-gap region size.

On the Fig. 26. only one fourth of the structure area is presented to show the borders of all elements of the reluctance network in study.

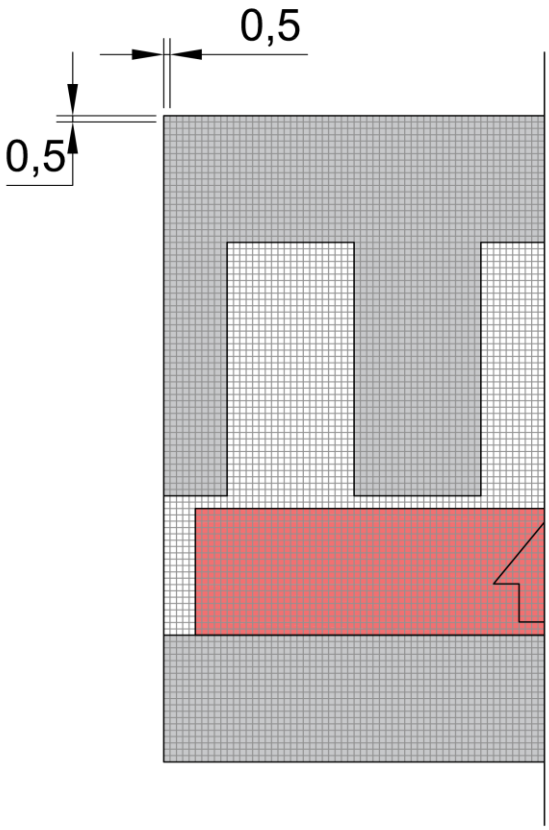

Fig. 26. Part of the linear structure with elements borders

The size of the smallest element here is 0.5 mm. As can be seen the RN perfectly fits this structure, which was done on purpose. However it is not always true for structures, which can be studied.

Coordinates for each node have to be calculated to create and cover the reluctance network over the structure. Therefore, formula for calculating coordinates for structures with complicated shape is more sophisticated than for one on Fig. 26. .

Same size for all reluctance elements allows not to specify reluctance value for each element separately, because all elements of steel will have the same reluctance. It is also correct for all elements of magnet and air. Therefore, only material index has to be assigned to each element. Doing this for the whole reluctance network manually is still a big job. However, there is an approach, which helps to simplify it. As can be noticed all elements inside, for example, left magnet area has the same material-index. Actually, a magnet is presented by a rectangle, thus the task is to determine coordinates, which are placed inside this rectangle and assign material-index of magnet material to them.

Basically, rectangle is a particular case of polygon. Thus, the same approach can be applied to all other elements of the structure, which are just more complicated polygons.

Fortunately, MatLab has a special function `inpolygon`, which works with this task. This function require coordinates of polygon as input and coordinates to be checked. This function has the following syntax

```
in = inpolygon(xq,yq,xv,yv)
```

This function returns `in` indicating if the query points specified by `xq` and `yq` are inside or on the edge of the polygon area defined by `xv` and `yv`. After the coordinates of nodes inside particular area (polygon) have been determined material-index can be assigned and then recalculated to corresponding reluctance value.

To initialize magnets, flux sources have to be placed into each vertical branch in magnets area. To do this special function has been developed, which at the first step gets coordinate of left-bottom node, right-top node and direction as input and results in number of branches which are in a rectangle formed by mentioned nodes and directed in specified direction. At the second step all these branches are initialized with flux sources.

After these steps all five vectors of input data are created. And CircuitAnalysis procedure can be run. It results in values of fluxes of all branches. And to be able to plot a radial component of flux density the particular branches have to be picked. This is done manually.

To validate correctness of the approach this structure was also modeled by FEM without taking saturation into account (as in reluctance method by default). Radial flux density component from FEM and reluctance network method were compared and results are presented on Fig. 27.

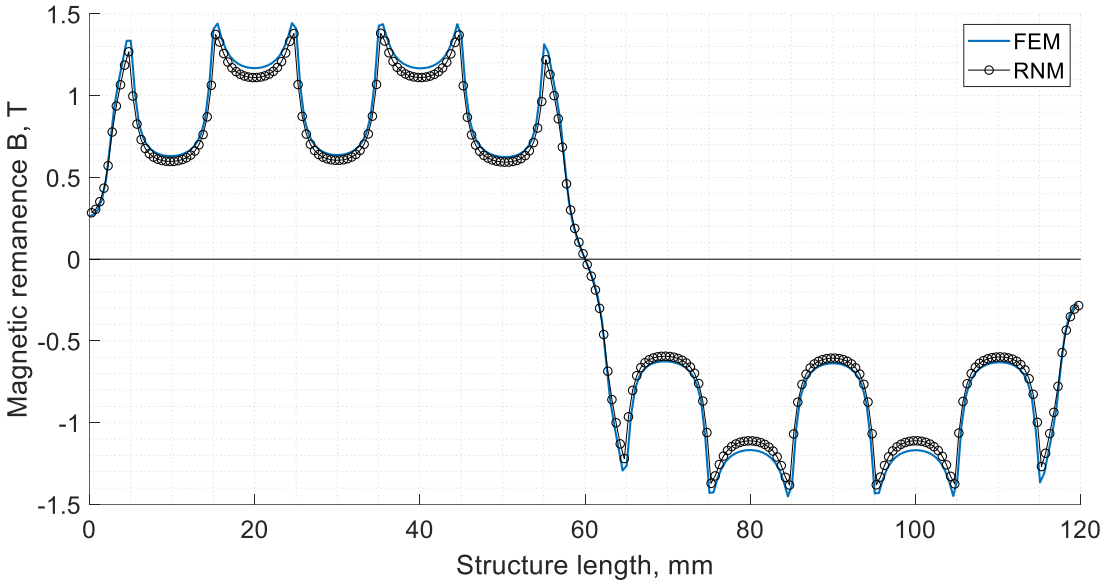

Fig. 27. Radial flux density component

As it can be seen, a good agreement between RNM and FEM is observed.

5 DISCUSSION AND CONCLUSIONS

The reluctance network method was studied in this paper and its application to the linear structure. The procedure for automatic circuit analysis was developed and its code is presented in the appendix. Without this procedure the method of equivalent circuit seems to be applicable only to small circuit, which is constructed only for the main flux path. Results have been validated by comparing with FEM as reference. Regardless, the fully automated CircuitAnalysis procedure the reluctance network method still require a lot of manual actions, number of which and its complexness only increase with the size of the studied structure. Therefore, further research related to this topic can include an attempt to develop new automation technics in addition to those, which were used in this thesis. Current approach also gives a possibility to take saturation into account, what was considered, but have not been implemented. Moreover, assumed constraint of equality of all elements sizes can also be reconsidered to make the approach more flexible. Finally this method can be applied to radial structures such as electrical machines. It will require to generate coordinates of reluctance network and polygons denoting different structure areas in a shape of circle.

In other words, the core of this thesis is fast procedure capable of solving circuits in a highly optimized way and a wrapper for particular linear structure. Other wrappers have to be developed to be able to use this procedure for other structures. This is the main direction of the future work.

REFERENCES

1. B.L.J. Gysen, K.J. Meessen, J.J.H. Paulides, E.A. Lomonova, General formulation of the electromagnetic field distribution in machines and
devices using Fourier analysis, IEEE Trans. Magn. 46 (2010) 39–52.

2. G. Male, T. Lubin, S. Mezani, J. Lévêque, Analytical calculation of the flux density distribution in a superconducting reluctance machine with ˆ
HTS bulks rotor, Math. Comput. Simulation 90 (2013) 230–243.

3. H. Tiegna, Y. Amara, G. Barakat, Overview of analytical models of permanent magnet electrical machines for analysis and design purposes,
Math. Comput. Simulation 90 (2013) 162–177.

4. D. Howe, Z.Q. Zhu, The influence of finite element discretisation on the prediction of cogging torque in permanent magnet excited motors,
IEEE Trans. Magn. 28 (1992) 1080–1083.

5. C.F. Iselin, Review of recent developments in magnet computations, IEEE Trans. Magn. 17 (1981) 2168–2177.

6. S. Shang, G.J. Yun, Stochastic finite element with material uncertainties: Implementation in a general purpose simulation program, Finite
Elem. Anal. Des. 64 (2013) 65–78.

7. T. Tortschanoff, Survey of numerical methods in field calculations, IEEE Trans. Magn. 20 (1984) 1912–1917.

8. G. Barakat, Y. Amara, A simple and effective way to couple analytical formal solution of magnetic potential and reluctance network models,
in: CEM2014, Ninth International Conference on Computation in Electromagnetics, London, 31 March–1 April, 2014.

9. A. Kladas, A. Razek, Analytic macroelements combined with finite elements: Application in modelling of electromagnetic systems, Math.
Comput. Simulation 28 (1986) 381–390.

10. A. Parviainen, M. Niemela, J. Pyrhonen, Modeling of axial flux permanent-magnet machines, IEEE Trans. Ind. Appl. 40 (2004) 1333–1340. ¨

11. J. Perho, Reluctance network for analysing induction machines, in: Acta Polytechnica Scandinavica, Espoo, in: Electrical Engineering Series,
no. 110, Finish Academies of Technology, 2002.
12. K.J.W. Pluk, J.W. Jansen, E.A. Lomonova, Hybrid analytical modeling: Fourier modeling combined with mesh-based magnetic equivalent
circuits, IEEE Trans. Magn. 51 (2015) paper 8106812.
13. D.A. Philips, Coupling finite elements and magnetic networks in magnetostatics, Internat. J. Numer. Methods Engrg. 35 (1992) 1991–2002.
14. Sofiane Ouagued, Yacine Amara, Georges Barakat, Comparison of hybrid analytical modelling and reluctance network modelling for pre-design purposes, 2016.
15. Ostovic V., Dynamic of Saturated Electric Machines. Springer-Verlag, New York, USA, 1989.
16. Demirchyan K.S., Neyman L.R., Korovkin N.V., Theoretical Electrical Engineering Moscow, St. Petersburg: Vol. 1, Piter, 2003. (In Russian).
17. By members of the staff of the Department of Electrical Engineering Massachusetts Institute of Technology, Principles of electrical engineering series. Magnetic Circuits and Transformers, The M.I.T. PRESS, 1962.
18. G. Zeveke, P. Jonkin, A. Netushil, S. Strakhov, Fundamentals of electrical circuit theory, Energoatomizdat, 1989. (In Russian)
19. William H. Hayt, Jr., Jack E. Kemmerly, Steven M. Durbin., Engineering circuit analysis, 8th ed., McGraw-Hill, 2012.
20. Incidence Matrix. WolframMathWorld. Available at http://mathworld.wolfram.com/IncidenceMatrix.html. Accessed in July, 2018.
21. Minimum spanning tree of graph - minspantree. Available at https://www.mathworks.com/help/matlab/ref/graph.minspantree.html. (MatLab 2018a) Accessed in July, 2018.
22. John R. Gilbert , Cleve Moler , Robert Schreiber, Sparse Matrices In MATLAB: Design And Implementation,1992.
23. MATLAB Data. Sparse Matrices. Available at https://www.mathworks.com/help/matlab/matlab_external/matlab-data.html. Accessed in July, 2018.

APPENDIX 1. Programm code

All the code related to studied structure is presented in next appendices. The last version of the CircuitAnalysis procedure also can be downloaded from GitHub repository https://github.com/Rostislaved/CircuitAnalysis. It allows me to timely maintain code and fix bugs to keep the newest version always available. Data from FEM simulation to be placed in FEM.txt is presented in appendix 21.

APPENDIX 2. Main.m

```
clear
close all
clc

tic
global la; % Active length, m
la = 1;
w = 120; % Structure width
h = 51; % Structure height
eW = 0.5; % Element Width
eH = 0.5; % Element Height
c = w / eW; % Number of elements in horizontal (columns)
r = h / eH; % Number of elements in vertical (rows)

[s, t] = formST(r, c, 0); % Form all branches
Rel = getReluctanceMatrix(s, t); % Node matrix of reluctances
Rel = reshape( Rel', 1,[]); % Node vector of reluctances
Rvec = Rel(s) + Rel(t); % Vector of branches reluctances
% Form Is ================================
r1 = 82;
r2 = 62;
fillValue = 1.2*0.5e-3;
Is = formSource(s, t, c);

% Form E =================================
E = zeros(size(Is));

% Solve ==================================
[ Nodes, Branches ] = CircuitAnalysis( s, t, E, Is,
Rvec,'method','nodal','order','nodes');

% Plot radial flux
mult = 0;
for i = 1:length(mult)
```

```
    s1 = (14401:14640)-mult(i)*c;
    t1 = (14641:14880)-mult(i)*c;

    A = [s1(:) t1(:)];
    B = Branches{:,2:3};
    [~,ind] = ismember(A,B,'rows');

    I(i,:) = Branches{ind,8}';
end

if 0 % Another representation og the plot
    yyaxis right
    h = line([0 w],[0 0],'linewidth',1.5,'Color','k');
    set(h,'tag','hline','handlevisibility','off')
    ylim([-1.6 2.24])
else
    figure('units','normalized','outerposition',[0     0.185     1
0.8157]) % Office
end
hold on

f = dlmread('FEM.txt');
n = length(f);
x = f(1:n,1);
y = f(1:n,3);

xx = linspace(0,n,n*2);
yy = interp1(x,y,xx);

plot(xx, yy, ...
    'LineWidth',1.5)

x = eW/2:eW:w-eW/2;
plot(x, I'/(la*eH),...
    'k',...
    'LineWidth', 0.5,...
    'Marker','o')
xlim([0 w])
axis0
cropImg
grid minor
ylabel('Magnetic remanence B, T')
xlabel('Structure length, mm')
legend('FEM','RNM')
set(gca,'FontSize',18)
toc
```

APPENDIX 3. formST.m

```
function [s, t] = formST(r, c, varargin)
% Forms two vectors describing branches which connect nodes in
% a rectangle r by n matrix:
% |       1       |      2       |      3       | ... |   c    |
% |      c+1      |     c+2      |     c+3      | ... |  2c    |
% |       .       |      .       |      .       |  .  |   .    |
% |       .       |      .       |      .       |  .  |   .    |
% |       .       |      .       |      .       |  .  |   .    |
% |   (r-1)*c+1   |  (r-1)*c+2   |  (r-1)*c+3   | ... |  r*c   |
%
% At first horisontal branches are formed:
%     . __________ . __________  . __________ . __________ .
%           1           5            9           13
%
%
%     . __________ . __________  . __________ . __________ .
%           2           6            10          14
%
%
%     . __________ . __________  . __________ . __________ .
%           3           7            11          15
%
%
%     . __________ . __________  . __________ . __________ .
%           4           8            12          16
% Then verical branches:
%     . __________ . __________  . __________ . __________ .
%     |     1      |      5     |      9     |     13     |
%   17|          18|          19|          20|          21|
%     |            |            |            |            |
%     . __________ . __________  . __________ . __________ .
%     |     2      |      6     |     10     |     14     |
%   22|          23|          24|          25|          26|
%     |            |            |            |            |
%     . __________ . __________  . __________ . __________ .
%     |     3      |      7     |     11     |     15     |
%   27|          28|          29|          30|          31|
%     |            |            |            |            |
%     . __________ . __________  . __________ . __________ .
%           4           8            12          16
%
%
% If there is a third argument in function and its value is 1
% then all nodes of the first column are connected
% to   all nodes of the last  column correspondingly.
```

```
s = [];
t = [];
% Horizontal edges
for i = 1:c-1
   s = [s ((1:r)-1)*c+(i+0)];
   t = [t ((1:r)-1)*c+(i+1)];
end

% Verctical edges
for i = 1:r-1
   s = [s (1:c)+c*(i-1)];
   t = [t (1:c)+c*(i-0)];
end


% connect last column?
if nargin == 3
    if varargin{1}
        for i = 1:r
           s = [s (i-1)*c+1];
           t = [t (i-1)*c+c];
        end
    end
end

s = s(:);
t = t(:);
```

APPENDIX 4. getReluctanceMatrix.m

```
function reluctanceMatrix = getReluctanceMatrixe(s, t)


w = 120;
h = 51;
eW = 0.5;
eH = 0.5;

c = w / eW;
r = h / eH;

%% Create Material Table
mu0 = 4*pi*1e-7;
mur = 7.5e3;
% la = 0.2; % Active length, m
global la;
Rs = (1/mu0/mur) * w/(h*la) /2;
Ra = (1/mu0/1  ) * w/(h*la) /2;
Rm = Ra;

MaterialTable = {0, Ra,'g'; ... % Air
                 1, Rs,'b'; ... % Steel
                 2, Rm,'r'; ... % Magnet
                 };
%% Set coordinated of polygons of different materials
% Left
% Steel
xS = [0 5 5 15 15 25 25 35 35 45 45 55 55 65 65 75 75 85 85 95 95
105 105 115 115 120 120 0 0];
yS = [0 0 20 20 0 0 20 20 0 0 20 20 0 0 20 20 0 0 20 20 0 0 20 20
0 0 30 30 0]+21;

xS2 = [0 120 120 0 0];
yS2 = [0 0 10 10 0];
% Magnet
xM = [2.5 57.5 57.5 2.5 2.5];
yM = [0 0 10 10 0]+10;

% Right
% Steel
% xSr = [0 4 4 15 15 25 25 35 35 45 45 55 55 60 60 0 0]+60;
% ySr = [0 0 20 20 0 0 20 20 0 0 20 20 0 0 30 30 0]+22;

% xS2r = [0 60 60 0 0]+60;
% yS2r = [0 0 10 10 0];
```

```
% Magnet
xMr = [2.5 57.5 57.5 2.5 2.5]+60;
yMr = [0 0 10 10 0]+10;


%% Create a structure of all polygons
Field1 = 'x';
Field2 = 'y';
Field3 = 'Material';
Value1 = {xS, xS2, xM, xMr};
Value2 = {yS, yS2, yM, yMr};
Value3 = {1, 1, 2, 2}; % 1 - Steel, 2 - Magnet, 3 - Air

area = struct(Field1, Value1, Field2, Value2, Field3, Value3);

%% Create coordinates of network nodes
% eW = w/c;
% eH = h/r;

x = [];
y = [];
for i = h-eH/2:-eH:eH/2
    for j = eW/2:eW:w-eW/2
        x = [x j];
        y = [y i];
    end
end

% Plot all polygons?
if 1
    figure('units','normalized','outerposition',[0     0.185     1
0.8157]) % Office
%          figure('units','normalized','outerposition',[0  0.017  1
0.983]) % Home


    for i = 1:length(area)
        Color    =    MaterialTable{find([MaterialTable{:,1}]    ==
area(i).Material),3};
        h = fill(area(i).x,area(i).y, Color,'LineWidth', 3);
        alpha(0.5)
        hold on
        set(h,'tag','hline','handlevisibility','off');
    end

    [s, t] = formST(r, c, 1);
    G = graph(s,t);
```

```
    p = plot(G,'XData', x,...
                'YData', y,...
                'NodeLabel',{},...
                'NodeColor','k',...
                'MarkerSize', 15*10/max(r,c),...
                'EdgeColor',[158 158 158]/255, ...
                'LineWidth', 0.5*10/max(r,c));
    set(p,'tag','hline','handlevisibility','off')
    cropImg
    hold off

    proc = 3;
    dx = ( max(x) - min(x) )/100 * proc;
    dy = ( max(y) - min(y) )/100 * proc;
    xlim([min(x) - dx max(x) + dx])
    ylim([min(y) - dy max(y) + dy])
end

% Add indices of (r,c) matrix
% related to material
% 0 - Air, 1 - Steel, 2 - Magnet.
for i = 1:length(area)
    idx = inpolygon(x, y, area(i).x, area(i).y);
    area(i).idx = find(reshape(idx,c,r)');
end

% Create reluctance indices matrix
reluctanceMatrixIdx = zeros(r,c);
for i = 1:length(area)
    reluctanceMatrixIdx(area(i).idx) = area(i).Material;
end

% Transform reluctanceMatrixIdx to reluctanceMatrix
Materials = unique(reluctanceMatrixIdx);
reluctanceMatrix = zeros(r,c);
for i = 1:length(Materials)
    ind  = find( reluctanceMatrixIdx == Materials(i) );
    ind2 = find( [MaterialTable{:,1}]  == Materials(i) );
    reluctanceMatrix(ind) = MaterialTable{ind2,2};
end
end
```

APPENDIX 5. formSource.m

```
function vec = formSource(s, t, c)
    vec = zeros(1,length(s));
    eH = 0.5;
global la;
    f = 1.2 * eH *la;
    lbNode = 19446;
    rtNode = 14995;
    Direction = 'd2u';
    fillValue = f;
    stNew = getST(c, lbNode, rtNode, Direction);
    vec   =   fillValues(vec,   s,   t,   stNew(1,:),   stNew(2,:),
fillValue);

    lbNode = 19566;
    rtNode = 15115;
    Direction = 'd2u';
    fillValue = -f;
    stNew = getST(c, lbNode, rtNode, Direction);
    vec   =   fillValues(vec,   s,   t,   stNew(1,:),   stNew(2,:),
fillValue);
end
```

APPENDIX 6. cropImg.m

```
function cropImg()
% Crop the image
    set(gca,'LooseInset',get(gca,'TightInset'))
end
```

APPENDIX 7. getST.m

```
function stNew = getST(c, lbNode, rtNode, Direction)
    rightCol = mod(rtNode,c);
    topRow = floor(rtNode/c) + 1;

    leftCol = mod(lbNode,c);
    bottomRow = floor(lbNode/c) + 1;

    % Area width and height
    areaW = rightCol - leftCol + 1;
    areaH = bottomRow - topRow + 1;

    % Create rectangle matrix of area
    for i = 1:areaH
        area(i,:) = rc2node(c, topRow + i - 1, rightCol-areaW +
1):rc2node(c, topRow + i - 1, rightCol);
    end

    % Make new s and t vectors describing desireble branches
    stNew = matrixCascade(area, Direction);


    function newM  = matrixCascade(M, direction)
        switch direction
            case 'u2d'
                % as it is
            case 'd2u'
                M = flipud(M);
            case 'l2r'
                M = rot90(M,-1);
            case 'r2l'
                M = rot90(M);
        end

        newM = [];
        for i = 1:size(M,1)-1
            newM =  [newM [M(i,:); M(i+1,:)]];
        end
    end

    function node = rc2node(c, currR, currC)
        % Convert currR and currC indices into linear index
        % of r x c matrix
        node = (currR-1)*c + currC;
    end
end
```

APPENDIX 8. fillValues.m

```
function vec = fillValues(vec, sDef, tDef, sNew, tNew, fillValue)
    sDef = sDef(:);
    tDef = tDef(:);
    sNew = sNew(:);
    tNew = tNew(:);
    vec  = vec(:);

    stDef = [sDef tDef];

    if (size(vec) ~= size(sDef))
        size(vec)
        size(sDef)
        error('size(vec) != size(sDef)');
    end

    idx = find(ismember(stDef, [sNew tNew], 'rows'));
    vec(idx, 1) = fillValue;

    idx = find(ismember(stDef, [tNew sNew], 'rows'));
    vec(idx, 1) = -fillValue;
    vec = vec';
end
```

APPENDIX 9. axis0.m

```
function axis0()

yl = ylim;
if (yl(1)<=0) & (yl(2)>=0)
    g = ishold;
    hold on

    xl = xlim;
    h = plot([xl(1) xl(2)],[0 0],'color','k');
    set(h,'tag','axis0','handlevisibility','off');  %  this  last
part is so that it doesn't show up on legends
    if g==0
        hold off
    end
end
end
```

APPENDIX 10. inc2nc.m

```
function An = inc2nc(B)
% Converts incidence matrix to nodes' conection matrix
An = -B(1:end-1,:);
end
```

APPENDIX 11. stSort.m

```
function [s, t, varargout] = stSort(s, t)
% Do not use with (di)graph functions! Order will be lost!
s = s(:);
t = t(:);

oldOrder = (1:length(s))';
EdgeTable = table([s t], oldOrder, 'VariableNames', {'EndNodes'
'Num'});
G = graph(EdgeTable);

mstG = minspantree(G);

newOrder1 = sort(mstG.Edges.Num);
newOrder2 = setdiff(oldOrder,newOrder1);
newOrder  = [newOrder1; newOrder2];
T  = EdgeTable{newOrder,1};
s = T(:,1);
t = T(:,2);
Tr  = EdgeTable(newOrder1,1);
Con = EdgeTable(newOrder2,1);

if nargout == 3
    varargout{1} = newOrder;
elseif nargout == 5
    G = digraph([Tr; Con]); % Full digraph
    T = digraph(Tr); % Digraph of minspantree

    varargout{1} = newOrder;
    varargout{2} = G;
    varargout{3} = T;
end

end
```

APPENDIX 12. CircuitAnalysis.m

```
function [Nodes, BranchesTable] = CircuitAnalysis( s, t, E, Is, R,
varargin)
%
==================================================================
======
% Circuit analysis function
%
==================================================================
======
% Inputs:
% 1. s  - Vector of start nodes
% 2. t  - Vector of target nodes
% 3. E  - Vector of branch emf
% 4. Is - Vector of branch current (in parralel)
% 5. R  - Vector of branch resistance
% 6 .varargin - name - value pair
%   6.1 Method of solving
%       Name: 'Method'
%       Value: 'Auto' (Default) | 'Nodal' | 'Mesh"
%   6.2 Disable computation of condition number because it can
take much time
%       Name: 'ConditionNumber'
%       Value: 'on' (default) | 'off'
%   6.3 Order of branches in Branches table (applies only for mesh
method)
%       Name: 'Order'
%       Value: 'Branches' (Defalut) | 'Nodes' | 'mst' |
%       'Branches' - order as was set in s and t vectors
%       'Nodes' - ascending order of nodes (sorted by algorithm
built in a 'graph' functuion)
%       'mst' - order as was sorted by convention for forming Mesh
matrix
%   6.4 Display matrix of coefficients in a symbolic way
%       Name: 'Coefficients'
%       Value: 'Hide' (Defalut) | 'Show'
%
==================================================================
======
% Outputs:
% 1. Nodes - Table with all information about nodes:
%   1.1 Num - Number of node
%   1.2 Phi - Node potential
% 2. Branches -  Table with all information about branches:
%   2.1 Num - Number of the branch
```

```
%   2.2 s - number of start node for branch Num
%   2.3 t - number of target node for branch Num
%   2.4 R - Branch resistance
%   2.5 E - Branch EMF
%   2.6 Is - Branch current (in parralel)
%   2.7 U - Branch voltage drop
%   2.8 I - Branch current (from node s to node t)
%
=================================================================
======

%% Prepare inputs
=============================================================
p = inputParse(varargin{:}); % Parse Name-value pair of arguments

s  = s(:);
t  = t(:);
E  = E(:);
Is = Is(:);
R  = R(:);
%
=================================================================
======

%%
=================================================================
=====
% Compute topological matrices of circuit and change order of
% branches: branches of tree first and branches of connections
next
[An, C, ~, newOrder] = topmat(s, t);

s  = s(newOrder);
t  = t(newOrder);
E  = E(newOrder);
Is = Is(newOrder);
R  = R(newOrder);

branchesNumbers = (1:length(s))';
branchesNumbers = branchesNumbers(newOrder);
%
=================================================================
======

%% Choose solving method
================================================
nodesMo     = size(An, 1); % Number of nodes minus one
```

```
indContours = size(C , 1); % Number of independent contours
switch p.Method
    case 'auto'
        if nodesMo < indContours % Nodal Analysis
            method = 1;
        else % Mesh Analysis
            method = 0;
        end
    case 'nodal'
        method = 1;
    case 'mesh'
        method = 0;
end
%
======================================================================
======

%% Computation by chosen method
=========================================
if method
    [I, U, phi, M] = NA(E, Is, R, An);
else
    [I, U, phi, M] = MA(E, Is, R, An, C);
end
%
======================================================================
======

%% Printing info
=========================================================
if method
    disp('- Nodal Analysis')
else
    disp('- Mesh Analysis')
end
disp(['- Your circuit has ' num2str( size(An, 1)+1 ) ' nodes and
'...
                           num2str( size(An, 2)   ) '
branches.'])

if isequal(p.ConditionNumber, 'on')
    tic
    fprintf(['- Condition number of coefficients matrix is ',
numFormat( condest(M) ), '\n']);
    time = toc;
    fprintf(['- Time for computing condition number is ',
numFormat( time ),' seconds\n']);
```

```
    % Rcond ================
    % 0 - badly conditioned
    % 1 - ill conditioned
    % Cond(est)=============
    % 0 - well  conditioned
    % 1e16 - ill conditioned
    % ======================
end

if isequal(p.Coefficients, 'show') % Display g - matrix?
    if method
        % Nodal
        Gs = sym('g',[1 size(An,2)]);
        Gs = diag(Gs);
        As = sym(An);
        Gss = As*Gs*As'
    else
        % Mesh
        r_d = sym('r',[1 size(C,2)]);
        r_d = diag(r_d);
        Cs = sym(C);
        Rs = Cs*r_d*Cs'
    end
end
%
======================================================================
======

%% Form output
============================================================
% Sort nodes
num1 = 1:length(phi)+1;
object1 = [ num1(:), [phi(:); 0] ];

% Form nodes table
Nodes = array2table(object1, 'VariableNames',
{'Num';'Potential'});

% Sort branches
Branches =  [branchesNumbers(:), s(:), t(:), R(:), E(:), Is(:),
U(:), I(:)];
switch p.Order
    case 'branches'
        [~, finalOrder] = sort(branchesNumbers);
    case 'nodes'
        EdgeTable = table([s,t], branchesNumbers, 'VariableNames',
{'EndNodes', 'Branch'});
```

```
        tempG = digraph(EdgeTable);
        [~, finalOrder] = ismember(tempG.Edges.Branch,
branchesNumbers);
    case 'mst'
        finalOrder = branchesNumbers;
end
Branches = Branches(finalOrder,:);

% Form branches table
BranchesTable = array2table(Branches, 'VariableNames',
{'Branch';'s';'t';'R';'E'; 'J'; 'U'; 'I'});
%
================================================================
======
```

APPENDIX 13. inputParse.m

```
function p = inputParse(varargin)
    p = inputParser;
    % Method
    defaultMethod = 'auto';
    validMethods = {'Auto', 'Nodal','Mesh'};
    checkMethod = @(x) any(validatestring(x,validMethods));

    % Compute condition number?
    defaultConditionNumber = 'on';
    validConditionNumber = {'on', 'off'};
    checkConditionNumber = @(x)
any(validatestring(x,validConditionNumber));

    % Show coefficients?
    defaultCoefficients = 'hide';
    validCoefficients = {'show', 'hide'};
    checkCoefficients = @(x)
any(validatestring(x,validCoefficients));

    % Branches order
    defaultOrder = 'Branches';
    validOrder = {'mst', 'Nodes', 'Branches'};
    checkOrder = @(x) any(validatestring(x,validOrder));


    addParameter(p,'Method',defaultMethod,checkMethod)

addParameter(p,'ConditionNumber',defaultConditionNumber,checkCondi
tionNumber)

addParameter(p,'Coefficients',defaultCoefficients,checkCoefficient
s)
    addParameter(p,'Order',defaultOrder,checkOrder)

    parse(p,varargin{:});

    p = structfun(@lower, p.Results,'UniformOutput',false);

end
```

APPENDIX 14. topmat.m

```
function [A, C, varargout] = topmat(s, t)
% A - connections matrix
% C - Mesh matrix
% Q (varargout{1}) - Sections matrix

[s, t, newOrder, g, T] = stSort(s, t);

% p = plot(g);
% Labels = 1:length(s);
% labeledge(p,s,t,Labels);
%
% highlight(p,T);


B = st2incidence(s,t);

A = inc2nc(B);
% sz = min(size(A));
sz = size(A, 1);
A1 = A(:,1:sz);
A2 = A(:,sz+1:end);
Q2 = inv(A1)*A2;

Q = [speye(sz) Q2];
C1 = -Q2';
C = [C1 speye(length(A)-sz)];

if nargout == 3
   varargout{1} = Q;
elseif nargout == 4
   varargout{1} = Q;
   varargout{2} = newOrder;
end

end
```

APPENDIX 15. stSort.m

```
function [s, t, varargout] = stSort(s, t)
% Do not use with (di)graph functions! Order will be lost!
s = s(:);
t = t(:);

oldOrder = (1:length(s))';
EdgeTable = table([s t], oldOrder, 'VariableNames', {'EndNodes'
'Num'});
G = graph(EdgeTable);

mstG = minspantree(G);

newOrder1 = sort(mstG.Edges.Num);
newOrder2 = setdiff(oldOrder,newOrder1);
newOrder  = [newOrder1; newOrder2];
T  = EdgeTable{newOrder,1};
s = T(:,1);
t = T(:,2);
Tr  = EdgeTable(newOrder1,1);
Con = EdgeTable(newOrder2,1);

if nargout == 3
    varargout{1} = newOrder;
elseif nargout == 5
    G = digraph([Tr; Con]); % Full digraph
    T = digraph(Tr); % Digraph of minspantree

    varargout{1} = newOrder;
    varargout{2} = G;
    varargout{3} = T;
end

end
```

APPENDIX 16. st2incidence.m

```
function B = st2incidence(s, t, varargin)
% Compute incidence matrix from s and t vectors
% Builtin function is not applicable because of
% inner sorting which is not possible to disable

s = s(:)';
t = t(:)';
vec = (1:max([s,t]))';

s1 = sparse(s) == vec;
t1 = sparse(t) == vec;

B = t1 - s1;
end
```

APPENDIX 17. inc2nc.m

```
function An = inc2nc(B)
% Just converts incidence matrix to nodes' conection matrix
An = -B(1:end-1,:);
end
```

APPENDIX 18. NA.m

```
function [I, U, phi, G] = NA(E, J, R, An)

E = sparse(E);
J = sparse(J);
R = sparse(R);

%     g  = sparse(1)./R;
%     g_d = diag(g);
%
%     G = An*g_d*An'; % Matrix of coefficients
%     E = E + R.*Is; % Transfrom current sourse in parralel branch
to emf sourse in serial
%     J = -sum((E.*g)'.*An,2); % Right part of the system
%     phi = G\J;   % Potentials from 1st to  n-1 nodes
%
%     U = An'*phi; % Potendials without taking sources into
account
%     U = (U + E); % Potendials with taking sources into account
%     I = U.*g;    % Currents

    g  = sparse(1)./R;
    g_d = diag(g);

    G = An*g_d*An'; % Matrix of coefficients
    J = An*(J - E.*g);  % Right part of the system (Currents)
    phi = G\J;   % Potentials from 1st to  n-1 nodes

    U = (phi'*An)'; % Branch voltage drop

    I = (U + E).*g;    % Currents

    phi = full(phi);
    I = full(I);
    U = full(U);

end
```

APPENDIX 19. MA.m

```
function [I, U, phi, rk] = MA(E, J, R, An, C)

E = sparse(E);
J = sparse(J);
R = sparse(R);

    r_d = diag(R);

    rk = C*r_d*C';  % Matrix of coefficients
    Ek = C*(E - R.*J);   % Right part of the system (Voltages)
    Ik = rk\Ek; % Solve system of linear equations

    I = (Ik'*C)' + J; % Vector of branch current

    U = I.*R - E; % Vector of branch voltage drop
    phi  = An'\U;

phi = full(phi);
I = full(I);
U = full(U);

End
```

APPENDIX 20. numFormat.m

```
function str = numFormat(value)

if value > 1000 | value < 0.1
    str = '%1.1e';
else
    str = '%1.1f';
end
str = sprintf(str,value);

end
```

APPENDIX 21. content of FEM.txt file

0.262	0.635	1.199	0.733	1.404	1.457	0.739	-0.630	-0.643	-1.169	-0.639	-1.188	-0.697	-1.163
0.263	0.633	1.194	0.719	1.371	1.453	0.760	-0.721	-0.647	-1.170	-0.638	-1.184	-0.687	-1.125
0.266	0.632	1.189	0.706	1.345	1.382	0.792	-0.791	-0.652	-1.171	-0.637	-1.181	-0.679	-1.080
0.272	0.631	1.186	0.697	1.319	1.284	0.826	-0.860	-0.657	-1.173	-0.637	-1.178	-0.672	-1.040
0.278	0.631	1.182	0.687	1.299	1.174	0.881	-0.917	-0.662	-1.175	-0.637	-1.176	-0.665	-0.994
0.288	0.630	1.179	0.680	1.282	1.072	0.940	-0.966	-0.669	-1.177	-0.637	-1.174	-0.660	-0.942
0.299	0.631	1.177	0.672	1.267	0.985	1.020	-1.010	-0.676	-1.180	-0.637	-1.172	-0.655	-0.882
0.314	0.631	1.174	0.667	1.254	0.921	1.112	-1.054	-0.685	-1.184	-0.638	-1.170	-0.651	-0.808
0.331	0.631	1.172	0.661	1.242	0.869	1.216	-1.097	-0.694	-1.187	-0.639	-1.169	-0.647	-0.737
0.354	0.632	1.171	0.657	1.232	0.830	1.316	-1.141	-0.705	-1.192	-0.641	-1.169	-0.644	-0.648
0.381	0.633	1.170	0.653	1.223	0.797	1.312	-1.179	-0.718	-1.196	-0.642	-1.168	-0.641	-0.567
0.414	0.635	1.169	0.650	1.215	0.772	1.307	-1.215	-0.733	-1.202	-0.644	-1.168	-0.639	-0.503
0.459	0.637	1.169	0.647	1.208	0.749	1.268	-1.268	-0.749	-1.208	-0.647	-1.168	-0.637	-0.458
0.504	0.639	1.168	0.644	1.202	0.733	1.215	-1.306	-0.772	-1.215	-0.650	-1.169	-0.635	-0.413
0.570	0.641	1.168	0.642	1.196	0.717	1.184	-1.311	-0.797	-1.223	-0.653	-1.170	-0.633	-0.381
0.650	0.644	1.169	0.641	1.192	0.705	1.135	-1.316	-0.831	-1.232	-0.657	-1.171	-0.632	-0.353
0.730	0.647	1.170	0.639	1.187	0.694	1.097	-1.217	-0.868	-1.242	-0.661	-1.172	-0.631	-0.332
0.814	0.651	1.170	0.638	1.184	0.685	1.056	-1.111	-0.922	-1.254	-0.667	-1.174	-0.631	-0.314
0.879	0.655	1.172	0.637	1.180	0.676	1.007	-1.021	-0.989	-1.267	-0.672	-1.176	-0.631	-0.299
0.943	0.660	1.174	0.637	1.177	0.669	0.964	-0.941	-1.074	-1.282	-0.680	-1.179	-0.630	-0.287
0.994	0.665	1.176	0.637	1.175	0.662	0.914	-0.882	-1.174	-1.299	-0.687	-1.182	-0.631	-0.278
1.042	0.672	1.178	0.637	1.173	0.657	0.857	-0.826	-1.290	-1.319	-0.697	-1.186	-0.631	-0.271
1.083	0.679	1.181	0.637	1.171	0.652	0.793	-0.792	-1.396	-1.345	-0.706	-1.189	-0.632	-0.266
1.124	0.687	1.184	0.638	1.170	0.647	0.715	-0.761	-1.439	-1.371	-0.719	-1.194	-0.633	-0.263
1.163	0.697	1.188	0.639	1.169	0.643	0.639	-0.738	-1.437	-1.404	-0.733	-1.199	-0.635	-0.262
1.205	0.708	1.192	0.640	1.168	0.640	0.544	-0.720	-1.444	-1.426	-0.751	-1.205	-0.637	
1.240	0.720	1.197	0.642	1.168	0.637	0.457	-0.704	-1.403	-1.454	-0.771	-1.211	-0.639	
1.273	0.736	1.203	0.644	1.167	0.634	0.387	-0.691	-1.380	-1.460	-0.795	-1.219	-0.642	
1.325	0.752	1.208	0.647	1.168	0.632	0.335	-0.680	-1.353	-1.438	-0.825	-1.227	-0.644	
1.361	0.775	1.216	0.650	1.168	0.630	0.283	-0.671	-1.330	-1.382	-0.862	-1.237	-0.648	
1.362	0.800	1.223	0.653	1.169	0.629	0.243	-0.663	-1.306	-1.252	-0.913	-1.247	0.652	
1.365	0.834	1.233	0.657	1.170	0.627	0.207	-0.657	-1.288	-1.137	-0.969	-1.260	-0.657	
1.261	0.872	1.243	0.661	1.171	0.626	0.177	-0.651	-1.272	-1.051	-1.051	-1.274	-0.663	
1.150	0.926	1.254	0.667	1.173	0.625	0.149	-0.646	-1.257	-0.968	-1.132	-1.292	-0.669	
1.055	0.992	1.268	0.672	1.175	0.625	0.123	-0.642	-1.245	-0.913	-1.262	-1.310	-0.676	
0.971	1.078	1.283	0.680	1.178	0.625	0.101	-0.638	-1.234	-0.861	-1.384	-1.332	-0.685	
0.909	1.179	1.300	0.687	1.181	0.625	0.079	-0.635	-1.225	-0.825	-1.439	-1.355	-0.695	
0.851	1.295	1.320	0.697	1.184	0.625	0.058	-0.633	-1.217	-0.795	-1.461	-1.381	-0.707	
0.815	1.401	1.345	0.706	1.188	0.625	0.038	-0.631	-1.209	-0.770	-1.455	-1.413	-0.721	
0.781	1.444	1.371	0.719	1.192	0.626	0.019	-0.629	-1.203	-0.750	-1.427	-1.440	-0.737	
0.758	1.442	1.404	0.732	1.197	0.627	0.000	-0.627	-1.197	-0.732	-1.404	-1.463	-0.758	
0.738	1.449	1.427	0.750	1.203	0.629	-0.019	-0.626	-1.192	-0.719	-1.371	-1.458	-0.781	
0.721	1.408	1.454	0.770	1.209	0.631	-0.038	-0.625	-1.188	-0.706	-1.345	-1.387	-0.815	
0.707	1.384	1.460	0.795	1.217	0.633	-0.058	-0.625	-1.184	-0.697	-1.320	-1.288	-0.851	
0.695	1.357	1.438	0.825	1.225	0.635	-0.079	-0.625	-1.181	-0.687	-1.300	-1.178	-0.908	
0.685	1.333	1.382	0.862	1.235	0.638	-0.101	-0.625	-1.178	-0.680	-1.283	-1.076	-0.971	
0.676	1.310	1.253	0.913	1.245	0.642	-0.123	-0.625	-1.175	-0.672	-1.268	-0.988	-1.054	
0.669	1.291	1.137	0.969	1.258	0.646	-0.149	-0.625	-1.173	-0.667	-1.254	-0.924	-1.150	
0.663	1.275	1.051	1.050	1.271	0.651	-0.177	-0.626	-1.171	-0.661	-1.243	-0.872	-1.260	
0.657	1.259	0.968	1.132	1.288	0.657	-0.208	-0.627	-1.170	-0.657	-1.233	-0.833	-1.365	
0.652	1.247	0.913	1.261	1.307	0.663	-0.243	-0.629	-1.169	-0.653	-1.223	-0.800	-1.363	
0.648	1.236	0.861	1.384	1.328	0.671	-0.284	-0.630	-1.168	-0.650	-1.216	-0.775	-1.361	
0.644	1.227	0.825	1.439	1.351	0.680	-0.336	-0.632	-1.168	-0.647	-1.208	-0.752	-1.324	
0.642	1.219	0.795	1.461	1.377	0.691	-0.387	-0.634	-1.167	-0.644	-1.203	-0.735	-1.273	
0.639	1.211	0.770	1.454	1.408	0.704	-0.460	-0.637	-1.168	-0.642	-1.197	-0.720	-1.245	
0.637	1.205	0.750	1.426	1.436	0.720	-0.545	-0.640	-1.168	-0.640	-1.192	-0.708	-1.198	