Algorithm for Optimal Dimensioning of Three Phase and Mono Phase Electric Power Lines Implemented in Java

CRISTIAN ABRUDEAN, MANUELA PANOIU
Electrical Engineering and Industrial Informatics Department
Polytechnic University of Timisoara, Engineering Faculty of Hunedoara
Revolutiei str. no 5, code 331128, Hunedoara
ROMANIA
{cristiana, m.panoiu}@fih.upt.ro http://fih.upt.ro/op/electro_comp.html

Abstract: - In this paper it was present a Java software package useful for dimensioning of low voltage lines mono phase and three phases (AC and DC). For electric grid modelling it was use tree type. This package software allows the estimation and calculation of parameters of power transmission lines of electric power. The software have a graphical user interface, so the user has the possibility to input of data and other characteristics of the electric line, passing to the calculation stage when the data are correctly and completely entered. It was use a recursive algorithm in order to calculate the total active and reactive currents and the admissible loss voltage based on the users (consumers) characteristics. The output of the results is shown on the screen.

Key-Words: - computer software, electrical grid, optimal dimensioning, Java

1 Introduction

The dimensioning of power transmission lines parameters of electric power has an essential importance in the design, construction and simulation of electric power systems. The results obtained are used for the studies of power flow, short circuit and stability. The design of transmission lines is often the key factor for the existence or absence of failures caused by lightning. The monitoring system of the high-voltage transmission tower is a new feature of the state maintenance for transmission lines [1]. It is designed to prevent the materials of transmission tower from being destroyed or stolen. Many engineering studies are usually performed by electric power utilities for the design of new transmission lines [1], [3], [6]. In this paper it was present a software package implemented in java useful for dimensioning of low voltage lines mono phase and three phases (ac and dc). The package is loaded with an easy-to-use graphical user interface that makes it suitable for the interactive input of the geometry and output of the parameters.

2. The electric networks modeling

It was use tree-type networks for modeling. There is a single node, root node, conventional numbering with 0. In this root node there is a voltage source assume ideal. There are not loop links. It is possible to connect 0 or many other nodes to a node and also 0 or many users.

The nodes are characterized by: connecting length between nodes, the conductor material, the isolation and the maximum temperature. It is possible to impose that the connections for a node to be made using a conductor with the same section like the one to be connected.

The users are characterized by active power, efficiency, admissible drop voltage in normal regime of functioning and in starting regime, the starting coefficient and the power factor (only in AC.).

The electrical networks are represented using trees and the dimensioning calculus is optimized for a minimum material consume.

The conditions that must be accomplished by the connections of nodes are:

- The node connection accomplishes the thermal criteria. This are the first applied criteria because by using this criteria a minimum section value are impose.
- The drop voltage in nominal functioning regime must be less than admissible value of whatever cup plying users.
- The start drop voltage must be less than admissible value for each user (It assume that one by one user are started

and the other ones remain in nominal functioning regime)

Because an electrical network is in generally the same values for some characteristic (i.e. temperature or material) it was provided the possibility to use defaults values – global for entire grid – for these.

The drop voltage for a section in mono phase ac or dc is:

$$\Delta U \cong 2 \cdot \rho \cdot \frac{l}{s} \cdot I_a + 2 \cdot x \cdot l \cdot I_r , \qquad (1)$$

Where ΔU is drop voltage (loss voltage), ρ is material resistivity, l = the conductor length, s is the conductor section, x is specific reactance, Ia is the active current and Ir is the reactive current (zero in DC).

The drop voltage for ac three phase (in case of all users being symmetrical)

$$\Delta U \cong \rho \cdot \frac{l}{s} \cdot I_a + x \cdot l \cdot I_r , \qquad (2)$$

where ΔU is the voltage loss on a phase.

The current for a user are:

in DC:

$$I = \frac{P}{\eta \cdot U} \qquad (3)$$

in AC single phase:

$$I_a = \frac{P}{\eta \cdot U} \cdot \cos\varphi ; \qquad (4)$$

$$I_r = \frac{P}{\eta \cdot U} \cdot \sin\varphi \qquad (5)$$

in AC three phase:

$$I_a = \frac{P}{3 \cdot \eta \cdot U} \cdot \cos\varphi ; \qquad (6)$$

$$I_r = \frac{P}{3 \cdot \eta \cdot U} \cdot \sin\varphi \qquad (7)$$

Using line voltages was avoided in order to obtain a model close to the mono phase ac model and for avoiding the calculus with $\sqrt{3}$.

3. The software package. The classes

3.1 ErrorReport

This class is use for exceptions report when using Java virtual machine. This is the usual situation. The class constructor use a parameter a `Throwable` object or a subclass and also a boolean variable. The boolean variable is use for decide if the application are terminated.

The `Throwable.printStackTrace()` method print the error message in a `PrintStream` object.

```
package pwgrid;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class ErrorReport extends
JFrame {
  public ErrorReport(Throwable thr) {
        this(thr, false);
    }
    public ErrorReport(Throwable thr,
boolean exitOnClose) {
       …
        }
    }
```

3.2 Stas

The class name suggests their destination. The class contains constants, arrays of constants, searching methods or interpolation methods. All methods and variables are static.

```
package pwgrid;
import java.util.*;

public class Stas {
  private Stas() {}
  public final static int TYPE_CU1 = 1,
TYPE_CU2 = 2…;
  public final static int IZO_60 = 1,
IZO_70 = 2;
public final static String[] IZO_NAMES =
new String[] {
  null, "Rubber 60Â°C", "PVC 70Â°C"};
public final static double RO_CU =
0.017, RO_AL = 0.03; //ohm*mm2/m
  private final static double[] section
= new double[] { 1.5, 2.5,…};
  //The admissible currents (thermical
criteria)
   private final static int[] cu1 = new
int[] {//Copper, 1 conductor dc
      26, 35,…};
   private final static int[] al1 = new
int[] {//Aluminium, 1 conductor dc
  …
   private final static int[] ctemp =
new int[] {//temperatures
   5, 10…};
//correction coeff 60Â°C
  private final static double[]
ctheta60 = new double[] {
  1.25, 1.19,…};
  …
```

```
  //calc.    correction    coeff    for
ambiental temperature
  private static double ctheta(int izo,
int tamb) {
   int  k  =  Arrays.binarySearch(ctemp,
tamb);
   if (k < 0) k = -(k+1);
   try {
     switch (izo) {
      case IZO_60: return ctheta60[k];
      case IZO_70: return ctheta70[k];
       }
   } catch(Exception e) {}
         return 0; }
  …
  //the   maximum   current   supported
public   static   double   getImax(int
sectIndex, int type, int izo, int tamb,
double da) {
  if ((sectIndex < 0) || (sectIndex >=
section.length)) return 0; //error
  double res = ctheta(izo, tamb);
  double dax = (da < 1) ? (0.875 /
Math.sqrt(da)) : 1;
  switch (type) {
   case TYPE_CU1: return …
   case …
  }}
//the section that support the maximum
current
public static double getSect(double max,
int type, int izo,int tamb, double da) {
  int k = getSectIndex(imax, type, izo,
tamb, da);
  return (k != -1) ? section[k] : 0;
}
//the standard section
public  static  int  getSectIndex(double
sect) {
   int k = Arrays.binarySearch(section,
sect);
  if (k < 0) k = -(k + 1); return k;
}
public   static   double   getSect(double
sect) {
   try {return
section[getSectIndex(sect)];}
catch(Exception e) {return 0;}
 }
   public static double getSectByIndex
(int index) {
      try   {return   section[index];}
catch(Exception e) {return 0;}
}
      public  static  int  getSectCount()
{return section.length;}
  }
```

3.3 Grid

The `Grid` class models a trees structure electric grid. The class contains inner classes `Grid.Node` for nodes and `Grid.Node.Consumer` for consumers (users) connected to nodes. It can be connect to each node zero or many consumers and also zero or many sub-nodes. The `Grid` class models a trees structure electric grid. The nodes objects contain the coupling lines characteristics (length, material, isolation). The consumers objects are defined using power, efficiency, power factor, starting coefficient and the admissible drop voltages in the normal regime and in the starting regime. The purpose of the used algorithm implemented in the `Grid` class is to obtain a minimum volume in condition of complying the restrictions impose by heating and drop voltages. Therefore – taking into account the trees structure of the grid – it was use recursive methods. Some of the characteristics of the nodes or consumers are often the same for the entire electric grid (materials, isolation, and the admissible drop voltages).

The inner classes can use references to the class that encapsulate them (i.e. in `Gride.Node` can be use `Grid.this`) They cannot be instantiate out of the `Grid` class. The `Grid` class contains a series of `set` and `get` methods use for private member access and default values for entire network. These values will be used for current values or for initialize new nodes or new users depending of some boolean variables. So, by example if the variable `etaImplValue = 0,8` and `isEtaImpl = true` then the efficiency of all consumers will be individually establish. If a new consumer will be created, the efficiency for this consumer will be η = 80%.

The `Grid.Node.precalc()` method calculate the total active and reactive currents and the admissible loss voltage based on the users (consumers) characteristics. Also, it determines a minimum section based on thermal criteria and based on a simplified admissible drop voltage criteria (it calculate the minimum section for obtaining the admissible drop voltage only for this node). The purpose of this method is to accelerate the calculus by limit the possible section set. There is a recursive method. The code for this method is presented here:

```
public void precalc() {
  iterationCount = 0;
  int k; double x;
```

```
  Ia = Ir = 0; DUamin = 1;
 for (k=0;k<getConsumerCount();k++) {
  Ia += getConsumer(k).getIa();
  Ir += getConsumer(k).getIr();
  if ((x= getConsumer(k).getDUa()) <
       DUamin) DUamin = x;
  }
  for (k=0;k < getSubNodeCount(); k++)
{
   Grid.Node snode = getSubNode(k);
   snode.precalc();
   Ia += snode.Ia; Ir += snode.Ir;
   if (DUamin > snode.DUamin) DUamin =
      snode.DUamin;
   }
  double I = Math.sqrt(Ia*Ia + Ir*Ir);
  minSectIndex = Stas.getSectIndex(I,
getStasType(), getIzo(),getTemp(),
Grid.this.DA);
  if (minSectIndex ==
     Stas.getSectCount()) return;
 x= getRo()*len*I/(DUamin*Grid.this.U);
 if (type != TYPE_AC3) x *= 2;
 k = Stas.getSectIndex(x);
 if (minSectIndex < k) minSectIndex = k;
  if (sectInherited && (parent != null)
&& (minSectIndex < parent.minSectIndex))
    minSectIndex = parent.minSectIndex;
  }
```

The `Grid.Node.calculate(double u, double r, double x)` method performed all calculus in order to obtain a minimum volume. The u parameter is the voltage node to which is made the connection. The parameters r and x are the cumulate resistance and reactance between node and source and are used for start checking's. The return value is the total volume for this section and for all sub-nodes, or the value 0 if the dimensioning criteria cannot be accomplished. The method search for a section that verified the dimensioning criteria and then increase the section until obtain 3 values in increasing order. Then the method return to the section for which was obtained the minimum volume. `Grid` and `Grid` inner classes implement `java.io.Serializable` interface used for load/save files. The code for this method is presented here:

```
public double calculate(double u, double
r, double x) {
 if(minSectIndex >= Stas.getSectCount())
return 0;
 int smin=minSectIndex, k, failcount=0;
 double v, vmin = 0, vol = 0, du, dr,
dx;
 boolean flag = true;
 double dup, dip;
 for (sectIndex = minSectIndex;
      sectIndex < Stas.getSectCount();
      sectIndex++)
    {
     iterationCount++;
     if (getSectInherited()) {
        sectIndex = parent.sectIndex;
        if (!flag) break;
        }
vol=Stas.getSectByIndex(sectIndex)*len;
    dr = getR(); dx = getX();
    du = dr*Math.sqrt(Ia*Ia + Ir*Ir);
    if  ((Grid.this.U  -  (u  -  du))  >
DUamin*Grid.this.U)
  {flag = false; continue;
        }
    flag = true;
  for(k=0;flag&&(k<getConsumerCount());
   k++)
  {dup=Grid.this.U-(u-du-(r+dr)*
  getConsumer(k).getDeltaIp());

if (dup > getConsumer(k).getDUap()
  *Grid.this.U)
flag = false;
    }
if (!flag) continue;
flag = true;
for (k = 0; flag && (k <
getSubNodeCount()); k++)
 {v=getSubNode(k).calculate(u-du, r+dr,
x+dx);
  if (v <= 0) flag = false;
  vol += v;
}
if (!flag) continue;
if ((vmin == 0) || (vmin >= vol)) {
failcount=0; vmin = vol; smin =
sectIndex;
}
   else failcount++;
if (getSectInherited()) break;
  // failcount is number of successive
//iterations for which was obtain a
//volume greater then minimum volume
if (Grid.this.getQuickAlgorithm() &&
(failcount >= 2)) break;
     }
if (vmin > 0)
  {sectIndex = smin;
  dr = getR(); dx = getX();
  du = dr*Math.sqrt(Ia*Ia + Ir*Ir);
  DU =(Grid.this.U-
   (u- du))/Grid.this.U;
  for (k=0; flag && (k <
        getSubNodeCount()); k++)
```

```
   {
getSubNode(k).calculate(u-du, r+dr,
x+dx); }
   for (k = 0; flag && (k <
    getConsumerCount()); k++)
   {dip = getConsumer(k).getDeltaIp();
    dup = 1-(u-du-
          (r+dr)*dip)/Grid.this.U;
    getConsumer(k).DUp = dup;
    }}
   return vmin;
   }
```

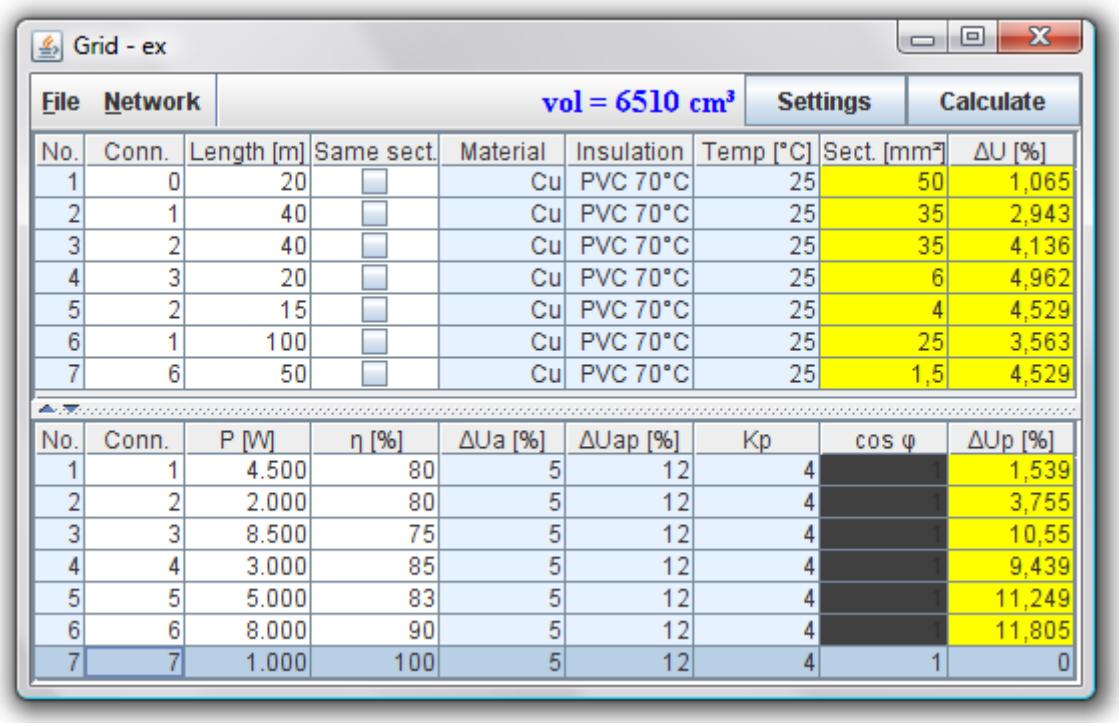
The Grid class offers also the possibilities to manage the nodes and the consumers. Therefore, it was used a series of methods for adding a node or a consumer and for deleting a node or a consumers.

```
public Grid.Node getSubNode(int index) {
    try
 {
   return(Grid.Node)subnodes.get(index);
 }
  catch(Exception e) {return null;
}
}
 public int getSubNodeCount() {
//the numbers of subnodes directly
connected here
  try {return subnodes.size();}
catch(Exception e) {return 0;}
  }

public int addSubNode() {
//return: subnode index
  Grid.Node node = new Grid.Node();
   return addSubNode(node);
}

public int addSubNode(Grid.Node node) {
//return: subnode index
  if (subnodes == null)
      subnodes = new Vector();
  subnodes.add(node);
  Grid.this.vnode.add(node);
  node.parent = this;
  return subnodes.size() - 1;
 }

public void removeSubNode(int index) {
//delete a subnode
 removeSubNode(getSubNode(index));
}

public void removeSubNode(Grid.Node
node) {
//delete a subnode
 if (node == null) return;
     vnode.remove(node);
}
```

```
public void moveSubNode(Grid.Node node,
Grid.Node newNode) {
  if ((node == null) ||
   (!subnodes.remove(node))) return;
if (newNode.subnodes == null)
  newNode.subnodes = new Vector();
  newNode.subnodes.add(node);
  node.parent = newNode;
 }
 //delete this node and all the subnodes
a consumers
public void delete() {
   int k;
   for (k = 0; k < getConsumerCount();
         k++) getConsumer(k).delete();
   for (k = 0; k < getSubNodeCount();
         k++) getSubNode(k).delete();
   if (parent != null)
         parent.removeSubNode(this);
  }

public Grid.Node.Consumer
             getConsumer(int index) {
   try {return  (Grid.Node.Consumer)
        consumers.get(index);}
   catch(Exception e) {return null;}
public int addConsumer() {
//return: consumers index
   if (consumers == null) consumers =
              new Vector();
   Grid.Node.Consumer con = new
           Grid.Node.Consumer(this);
   consumers.add(con);
   Grid.this.vcons.add(con);
   return consumers.size() - 1;
 }

public Grid.Node.Consumer addConsumer2()
{ int k = addConsumer();
   return getConsumer(k);   }
 public  void  removeConsumer(int  index)
{//delete consumer
   Grid.Node.Consumer con =
              getConsumer(index);
  if (con == null) return;
  consumers.removeElementAt(index);
  Grid.this.vcons.remove(con);
}

public void moveConsumer (
Grid.Node.Consumer con, Grid.Node
newNode) {
   if ((con == null) ||
     (!consumers.remove(con))) return;
  if (newNode.consumers == null)
       newNode.consumers = new Vector();
  newNode.consumers.add(con);
 con.node = newNode;
}
```

3.4 MainFrame

This class are derived from `javax.swing.JFrame` and represent the graphical user interface (GUI) for the software package. The constructor for this class use an argument an `Grid` object. This object will be data source for GUI. The GUI are depicted in fig. 1.



No.	Conn.	Length [m]	Same sect.	Material	Insulation	Temp [°C]	Sect. [mm²]	ΔU [%]
1	0	20	☐	Cu	PVC 70°C	25	50	1,065
2	1	40	☐	Cu	PVC 70°C	25	35	2,943
3	2	40	☐	Cu	PVC 70°C	25	35	4,136
4	3	20	☐	Cu	PVC 70°C	25	6	4,962
5	2	15	☐	Cu	PVC 70°C	25	4	4,529
6	1	100	☐	Cu	PVC 70°C	25	25	3,563
7	6	50	☐	Cu	PVC 70°C	25	1,5	4,529

No.	Conn.	P [W]	η [%]	ΔUa [%]	ΔUap [%]	Kp	cos φ	ΔUp [%]
1	1	4.500	80	5	12	4	1	1,539
2	2	2.000	80	5	12	4	1	3,755
3	3	8.500	75	5	12	4	1	10,55
4	4	3.000	85	5	12	4	1	9,439
5	5	5.000	83	5	12	4	1	11,249
6	6	8.000	90	5	12	4	1	11,805
7	7	1.000	100	5	12	4	1	0

Fig. 1. The main window of the software package

By using the "Calculate" button the dimensioning calculus will be made. For the electrical network from fig. 2, the calculus is performed in approx. 0.005 s. If the network is deep with many nodes and branches, the time need for computing may be longer (seconds or even minutes). The "Calculate" button executes the algorithm for dimensioning the selected grid. In the "Network" menu the user can select if the calculus will be performed with the "Fast algorithm" option. This can be necessary if the network is deep with many nodes and branches, because the recursive methods used for calculus. These methods was presented in section 3.3.

In the "Network" menu the user can also select the type of the electric grid, like in figure 3.

The inner class `VolumeLabel` is derived from `javax.swing.`JLabel and used for total volume displaying using a mark out format.

The inner class `SerializationFileFilter` is derived from `javax.swing.filechooser.FileFilter` and used for file filters in Open/Save file dialogs, form "File" menu. This menu contains the usual options: New, Open, Save, Save as … and Exit. Using this options the user can save a network, open a saved network or create a new network. The type of files are *.ser (serialization files).

`OptionsFrame` are a separate window and can be close/open using "Settings" button. This window is show in fig. 4. The second column contains checkbox controls for implicit values.

The inner class `ColorCellRender` is derived from `DefaultTableCellRenderer` (from package `javax.swing.table`). This class is used for modifying the swing component used for displaying some cell tables. The cells corresponding to default values (cannot modifying from the main window) are display on a grey-blue background. The cells corresponding to the computed results are displayed on a yellow background. The cells displayed on a dark gray background are irrelevant for the modelled network (i.e. cos φ in DC). If a cell value are identify to be a floating point number, will be displayed with maximum 3 decimals.

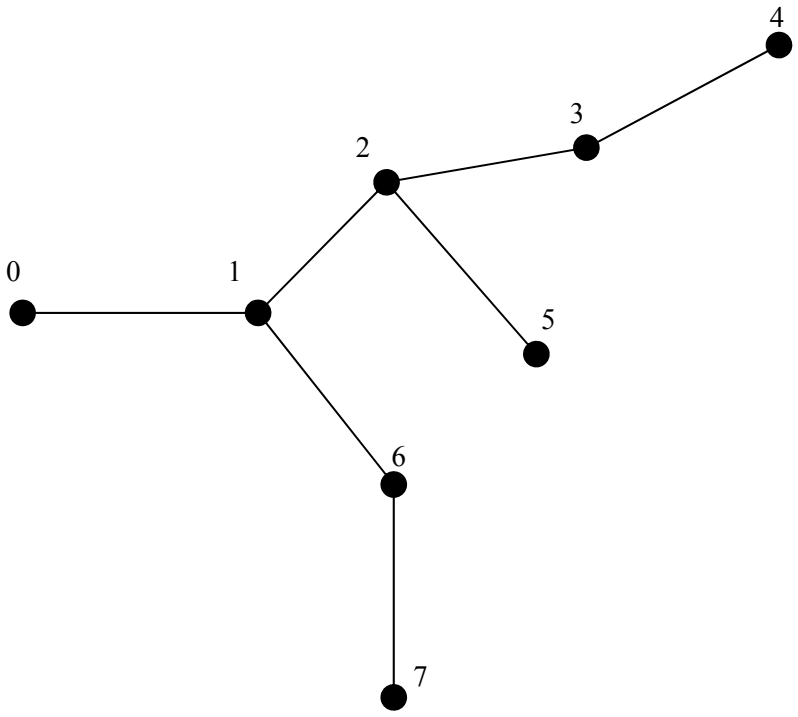


Fig. 2. An example of tree associated to a network

Grid - ex

File | Network | vol = 6510 cm³ | Settings | Calculate

- Direct Current
- Single Phase
- Three Phase
- Fast Algorithm

No.			ame sect.	Material	Insulation	Temp [°C]	Sect. [mm²]	ΔU [%]
1				Cu	PVC 70°C	25	50	1,065
2				Cu	PVC 70°C	25	35	2,943
3				Cu	PVC 70°C	25	35	4,136
4				Cu	PVC 70°C	25	6	4,962
5				Cu	PVC 70°C	25	4	4,529
6	1	100		Cu	PVC 70°C	25	25	3,563
7	6	50		Cu	PVC 70°C	25	1,5	4,529

No.	Conn.	P [W]	η [%]	ΔUa [%]	ΔUap [%]	Kp	cos φ	ΔUp [%]
1	1	4.500	80	5	12	4	1	1,539
2	2	2.000	80	5	12	4	1	3,755
3	3	8.500	75	5	12	4	1	10,55
4	4	3.000	85	5	12	4	1	9,439
5	5	5.000	83	5	12	4	1	11,249
6	6	8.000	90	5	12	4	1	11,805
7	7	1.000	100	5	12	4	1	0

Fig. 3. Selecting the type of network and the type of algorithm

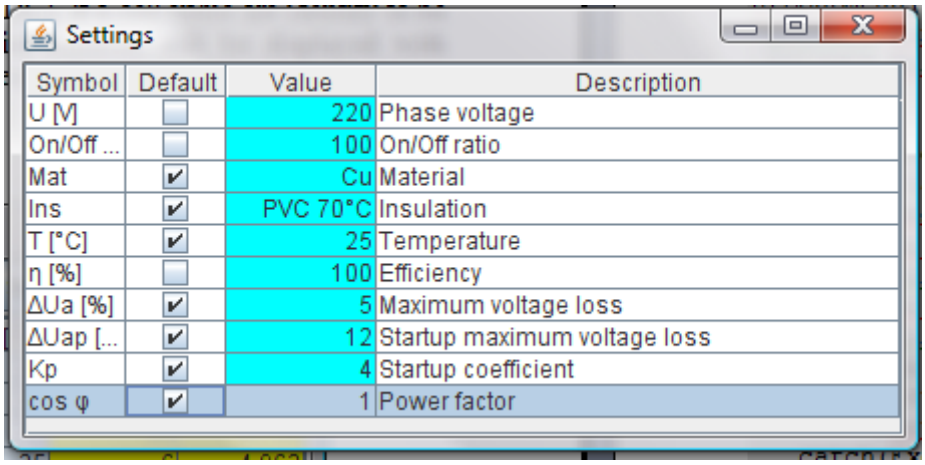


Fig. 4 The Optionframe window

The tables (`JTable`) are associate with data models being a handler between table and dates that can be displayed or modify. This class (`OptionsFrame`) contains two inner classes (`OptionTable` and `OptionTableModel`).

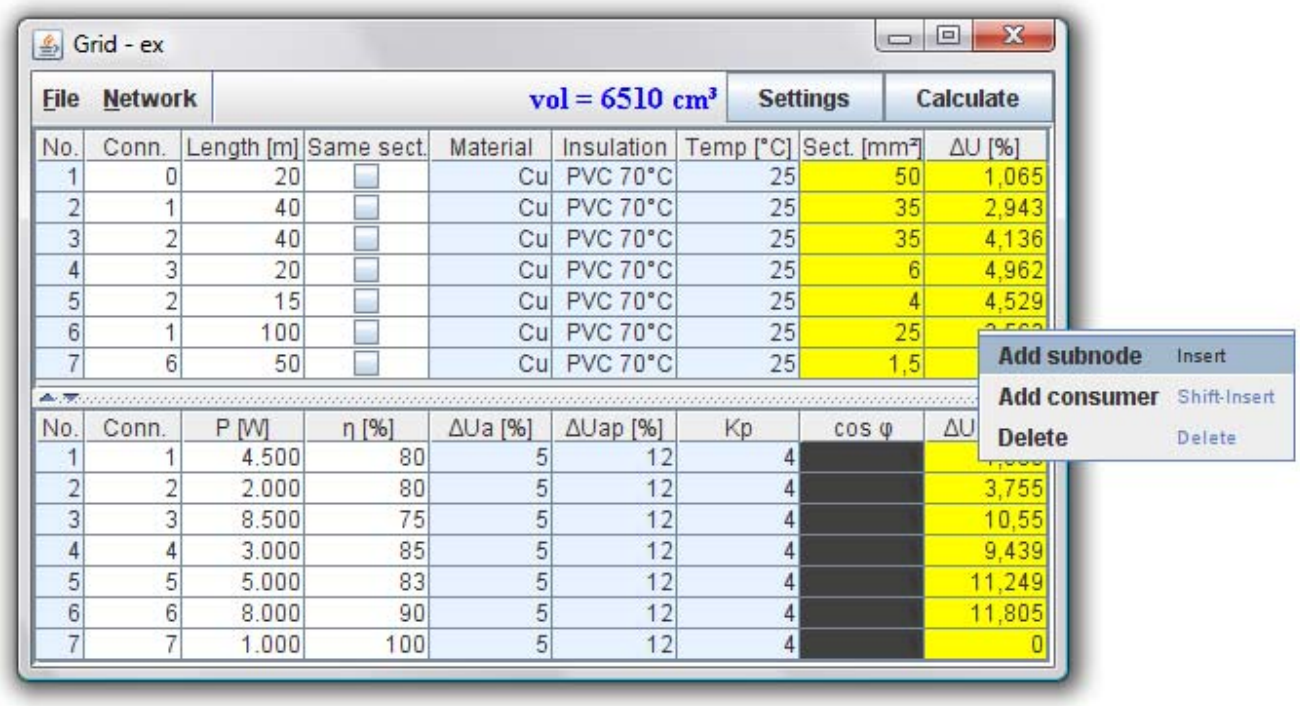
`NodeTable` implement the table of nodes. Row 3 from fig. 1 can be interpret as follow: "the node 3 are connected with the node 2 using a cable with length 40 m, the cable material are copper, isolation PVC, the functioning temperature 25°C. The results are: section 35 mm^2 and a nominal voltage 4,163 %".

The model for this table is an inner class. The model communicate with the instance for class `Grid` transmitted by `MainFrame` constructor. This class is presented here:

```
private class NodeTable extends JTable {
 private JPopupMenu popup = new
JPopupMenu();
 private int mouseRow = 0;
 private MainFrame.NodeTable.Model
model;
    ...
  // The model for the nodes table
  public class Model extends
AbstractTableModel {
 public int getColumnCount() {
     return colNames.length;}
 public int getRowCount() {
     return grid.getNodeCount();}
public Class getColumnClass(int c) {
    try {return getValueAt(0, c)
        .getClass();}
   catch(Exception e) {return
              Object.class;
}
}
public Object getValueAt(int row, int
col) {
 switch (col) {
  case COL_NR:return new Integer(1+row);
  case COL_CONEX:
          …
   }
public void setValueAt(Object obj,
                  int row, int col) {
        ...
    }
 public boolean isCellEditable(int row,
int col) {
       switch (col) {
        ...
  }
public String getColumnName(int col)
       {return colNames[col];}
        }

// The constructor NodeTable
      public NodeTable() {
         super();
         model = new
       MainFrame.NodeTable.Model();
         super.setModel(model);
          ...
     }
        ...
    }
```

`ConsTable` implements the users (consumers) table. Row 6 from fig 1 can be read like this: "The users 6 are connected to node 6, this utile power is 8000W, the efficiency 90 %, the admissible loss voltage 5%, the admissible loss voltage on start is 12 %, the start coefficient 4 , cos φ = 1 (irrelevant in DC). Results a loss voltage on starting 11,805%.

Both `NodeTable` and `ConsTable` contain a inner class `PopupListener`, used for open popup menus. Such a menu is show in fig. 4.



No.	Conn.	Length [m]	Same sect.	Material	Insulation	Temp [°C]	Sect. [mm²]	ΔU [%]
1	0	20		Cu	PVC 70°C	25	50	1,065
2	1	40		Cu	PVC 70°C	25	35	2,943
3	2	40		Cu	PVC 70°C	25	35	4,136
4	3	20		Cu	PVC 70°C	25	6	4,962
5	2	15		Cu	PVC 70°C	25	4	4,529
6	1	100		Cu	PVC 70°C	25	25	
7	6	50		Cu	PVC 70°C	25	1,5	

No.	Conn.	P [W]	η [%]	ΔUa [%]	ΔUap [%]	Kp	cos φ	ΔU
1	1	4.500	80	5	12	4		
2	2	2.000	80	5	12	4		3,755
3	3	8.500	75	5	12	4		10,55
4	4	3.000	85	5	12	4		9,439
5	5	5.000	83	5	12	4		11,249
6	6	8.000	90	5	12	4		11,805
7	7	1.000	100	5	12	4		0

Fig. 5. Adding and deleting nodes or consumers

From this popup menu it can be add or delete a node or a consumer. If the user delete a nod will be deleted also the all subnodes and all the consumers connected to this node. For adding or deleting a subnode it was used recursive methods, presented in section 3.3.

The table `ConsTable` offer a single option – for deleting the selected consumer. The described tree from the previous example is depicted in fig. 2.

The nodes numbering cannot be controlled through the interface. The assigned nodes numbers (also consumer's numbers) describe the introducing order only. For avoiding errors, the numbering must respect the following rule: when a number is assigned to a new node it must be connection between the new node ant the tree root. In the tree from fig. the node 3 cannot be introduce before the nodes 1 and 2.

4 Conclusion

This software package is very useful for electrical grid dimensioning for single phase and three phase electrical grids and also for DC and AC. The software can be also useful as educational software for students in electrical engineering.

References:

[1] Ren, Lijia, Jiang, Xiuchen, Sheng, Gehao, Bo, Wu Design and calculation method for dynamic increasing transmission line capacity, *WSEAS Transactions on Circuits and Systems*, vol 7, n 5, May, 2008, p 348-357.

[2] Grivet-Talocia, S., Salio, S.; Canavero, F. PULP, an educational software package for line parameters calculation IEEE *International Symposium on Electromagnetic Compatibility*, 1997, p 438-441

[3] Electric dimensioning criteria of 750 kV overhead lines Cristovici, A. (ISPE, Bucharest, Romania); Popescu, A.; Vatra, F. Source: *Energetica*, v 34, n 7, July 1986, p 322-4

[4] Carton, T.; Cerisier, J.; Lapeyre, J.L.; Vieille, J. CAMELIA: a calculation program for mechanical dimensions of MV, LV overhead lines Source: *IEE Conference Publication*, n 297, 1988, p 141-145

[5] Gustavo Rodríguez, Luis Aromataris , Marcos Donolo, Jose Hernandez and Diego MoitreSoftware for calculation of electric power systems parameters, *Computers & Electrical Engineering*, Vol. 29, Issue 5, July 2003, Pages 643-651

[6] Ekonomou, L., Fotis, G.P., Maris, T.I., Cost related optimum design method for overhead high voltage transmission lines, *European Transactions on Electrical Power*, v 18, n 5, July, 2008, p 437-447

[7] Lijia Ren, Xiuchen Jiang, Gehao Sheng, Wu Bo A New Study in Maintenance for Transmission Lines, *WSEAS TRANSACTIONS on CIRCUITS AND SYSTEMS*, Issue 5, Volume 7, May 2008, pag 392-401

[8] Carmen Escribano, Iglesias Antonio, Giraldo Carbajo María, Asunción Sastre Rosa, Interactive tools for Discrete Mathematics e-learning, *WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION*, Issue 2, Volume 5, February 2008, pp 97-103

[9] Drosopoulos A, Hatziprokopiou M, Hatziprokopiou E, Training Engineers for Power Line Communications, 5th *IASME/WSEAS International Conference on Engineering Education*, JUL 22-24, 2008 Heraklion, GREECE, Pages: 366-371 , 2008

[10] D'Ottavi S, Muzi F, Passacantando L, A real-time prediction procedure of tine state of an electrical distribution system, *6th WSEAS International Conference on Applications of Electrical Engineering* Istanbul, TURKEY, MAY 27-29, 2007, Pages**:** 241-245