A Modelica model of thermal and mechanical phenomena in bare overhead conductors

Oscar Duarte
Universidad Nacional de Colombia, Department of Electrical and Electronics Engineering
ogduartev@unal.edu.co
Carrera 30, Calle 45, Ed. 453, Of. 202. Bogotá, Colombia

Abstract

A Modelica model of the thermal and mechanical phenomena of bare overhead conductors is presented. The geometry of the catenary is also modeled as an important part of the mechanical phenomena. The model has been compiled and tested using OpenModelica. Some application examples are also presented to illustrate some analysis that can be done with the model.

Keywords: overhead conductors, IEEE 738, catenary, sag

1 Introduction

Bare overhead conductors are essential in transmission and distribution of large amounts of electrical energy. They are supported by transmission towers and exposed to varying weather conditions. The mechanical behavior of these suspended cables is affected by thermal processes. Heat transfers change conductor temperature, causing lenght and tension modifications. As a result, the geometry of the catenary described by the cable also changes. It is very important to study this geometric modifications, mainly to avoid electrical failures caused by the violation of security distances.

In this paper we show a Modelica model of the thermal and mechanical phenomena of bare overhead conductors; the geometry of the catenary is also modeled as an important part of the mechanical phenomena. The model has been compiled and tested using OpenModelica ([2], [3]). Some application examples, and the source code are available at the Virtual Academic Services of the National Universty of Colombia [1].

[1]http://www.lab.virtual.unal.edu.co

By using Modelica two main advantages have arised:

- The solution of the state change equation is very simple. This is an equation that must be solved by numerical methods. The Modelica model of the equation is done in a natural way, and the algorithms available in OpenModelica are capable to solve it.
- The cargability analysis is immediate. The most common models from overhead conductors use the electrical current in the conductor to compute the conductor temperature for extreme operation conditions and then the mechanical and geometric variables. Cargability analysis is the inverse process: from the mechanical, geometric or thermal limit conditions we must find an electrical current that will cause them. Due to the object-oriented modeling of Modelica, the same model can be used in both directions.

This paper is organized as follows: in section 2 the physcal model is presented in two steps, the thermal model first (section 2.1) and then the mechanical model (section 2.2); in section 3 the Modelica implementation is explained also in two steps (sections 3.1 and 3.2); in section 4 we show some application examples to illustrate the analysis capabilities of the implementation. Conclusions and future work are summarized in section 5.

2 Physical model

2.1 Thermal model

The thermal model calculates the conductor temperature for a certain electrical current and weather conditions. The IEEE Standar 738 ([1])

defines two different models: one for steady state calculations and another one for transient calculations.

The non-steady heat balance is summarized by equation 1

$$\frac{dT_c}{dt} = \frac{1}{mC_p}\left[RI^2 + q_s - q_c - q_r\right] \tag{1}$$

Where:

- T is the conductor temperature.
- mC_p is the heat capacity of conductor.
- R is the electrical linear resistence of the conductor, which is a function of its temperature.
- I is the current passing through the conductor. The term RI^2 is the heat gain by Joule effect.
- q_s is the heat gain by solar radiation.
- q_c is the heat loss by convection.
- q_r is the heat loss by radiation.

The standard also stablishes detailed models for every term in equation 1. The main features of these models are:

R **model:** electrical linear resistence of conductor is calculated as a function of temperature by a linear interpolation (or extrapolation) of the values at two different temperatures (usually 25°C and 75°C). These values are available in manufacturers data sheets.

mCp **model:** total heat capacity is calculated taking into account the percentage of materials in the conductor (ussualy steel and alluminium).

q_s **model:** the actual heat gain by solar radiation depends on the geographical latitude and altitude in which the conductor is placed, the day of the year, the time of the day, the abosorvity and size of conductor; two atmosphere conditions are considered: clear and industrial.

q_c **model:** natural and forced convection must be calculated. The greatest of the two is used in equation 1. The wind velocity and direction is considered; air density and viscosity depends on air temperature. Temperature, size and azimuth of conductor are also involved in the model.

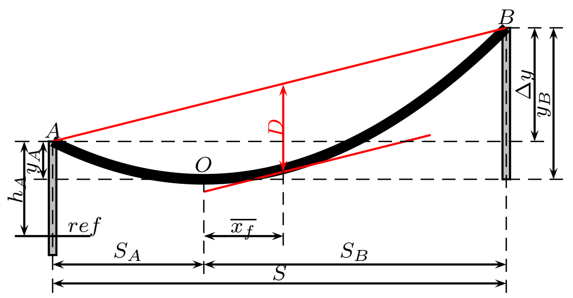

Figure 1: Geometry of the catenary

q_r **model:** radiated heat is calculated using temperature, emmisivity and size of conductor as well as air temperature.

Notice that T, which seems to be explicity calculated in 1 is really involved in all the terms. A more accurate equation shoud be:

$$\frac{dT}{dt} = \frac{1}{mC_p}\left[R(T)I^2 + q_s(T) - q_c(T) - q_r(T)\right] \tag{2}$$

2.2 Mechanical model

Mechanical phenomena in overhead conductors are well known (see for example chapter 14 of [4]). Figure 1 shows an overhead conductor that describes a catenary. It is supported in A y B, which are at height y_A and y_B from the level of the lowest point (O), and with a difference of level Δy. h_A is the height of A from a reference level. The span (horizontal separation between supports) is S. The longitudinal tension is Ten, where as the horizontal tension is H. Conductor lenght is L, its linear weight is W and is temperature is T. The lowest point O is at a horizontal distance S_A and S_B from supports A and B. The length of conductor from the supports A and B to the lowest point O are L_A and L_B.

Sag D is the maximum vertical distance between the imaginary line that connects A and B and the catenary. Sag occurs at the point in which the catenary tangent is the same as the slope of that imaginary line. The horizontal distance between this point and the lowest point O is $\overline{x_f}$.

2.2.1 Geometric considerations

The height of the catenary $y(\overline{x})$ from the level of the lowest point O is:

$$y(\overline{x}) = \frac{H}{W}\cosh\left(\frac{W\overline{x}}{H}\right) - \frac{H}{W}; \quad (3)$$

Where $\overline{x}$ is the horizontal distance to O. Let x be the horizontal distance to A. Then we have:

$$\begin{array}{rcl} \overline{x} & = & S_A - x \\ y(x) & = & \frac{H}{W}\cosh\left(\frac{W(S_A-x)}{H}\right) - \frac{H}{W} \end{array} \quad (4)$$

The height of the conductor in x from the reference level is

$$h(x) = h_A - Y_A + y(x) \quad (5)$$

$$\begin{array}{rl} h(x) = & h_A - \frac{H}{W}\cosh\left(\frac{WS_A}{H}\right) + \\ & \frac{H}{W}\cosh\left(\frac{W(S_A-x)}{H}\right) \end{array} \quad (6)$$

The horizontal distance from the lowest point O to the lowest support is S_{PB}

$$S_{PB} = \frac{S_A}{2} - \frac{H}{W}\sinh^{-1}\left\{\frac{\Delta y/2}{\frac{H}{W}\sinh\left(\frac{W}{H}S_A/2\right)}\right\} \quad (7)$$

Notice that S_{PB} is equal to S_A or S_B :

$$S_A = \begin{cases} S_{PB} & \text{if } \Delta y \geq 0 \\ A - S_{PB} & \text{if } \Delta y < 0 \end{cases} \quad (8)$$

2.2.2 Computing the sag

In order to find the sag, first we find the slope m of the imaginary line that connects A and B:

$$m = \frac{\Delta y}{S} \quad (9)$$

The first derivative of equation 3 give us the catenary tangent:

$$\frac{dy}{d\overline{x}} = \sinh\left(\frac{W\overline{x}}{H}\right) \quad (10)$$

$\overline{x_f}$ s the point where the slops equals m, so we have:

$$\overline{x_f} = \frac{H}{W}\mathtt{asinh}\left(\frac{\Delta y}{S}\right) \quad (11)$$

The sag D is the vertical distance from the imaginary line that connects A and B, y_r, and the catenary y_c, both in $\overline{x_f}$

$$\begin{array}{rcl} D & = & y_r(\overline{x_f}) - y_f(\overline{x_f}) \\ y_r(\overline{x_f}) & = & \frac{H}{W}\cosh\left(\frac{WS_B}{H}\right) - \frac{H}{W} - m(S_b - x_f) \\ y_c(\overline{x_f}) & = & \frac{H}{W}\cosh\left(\frac{W\overline{x_f}}{H}\right) - \frac{H}{W} \end{array} \quad (12)$$

2.2.3 Computing the horizontal tension

The total lenght L of the cable can be computed from $L = L_A + L_B$

$$L_A = \frac{H}{W}\sinh\left(\frac{WS_A}{H}\right) \qquad L_B = \frac{H}{W}\sinh\left(\frac{WS_B}{H}\right) \quad (13)$$

The cable longitudinal tensión at a distance $S/2$ from O is

$$Ten = H\cosh\frac{WS}{2H} \quad (14)$$

Suppose two different cable states 0 y 1. There are now two longitudinal tensions Ten_0 and Ten_1, two horizontal tensions H_0 and H_1, two temperatures T_0 and T_1 and the two lenghts L_0 and L_1. Then we have the *state change equation*:

$$L_1 = L_0\left[1 + a(T_1 - T_0) + \frac{Ten1 - Ten0}{EA}\right] \quad (15)$$

Where a is the coefficient of dilatation, E the elasticity module and A the section area. Notice that the new length must also satisfy equation 13. Usually numerical methods must be used to satisfy simultaniously equations 13 and 15.

Assuming that W does not change from state 0 to state 1, the value of H_1 is obtained by the solution of

$$\begin{array}{l} \frac{H_1}{W}\sinh\left(\frac{WS_A}{H_1}\right) + \frac{H_1}{W}\sinh\left(\frac{W(S-S_A)}{H_1}\right) = \\ \left\{\frac{H_0}{W}\sinh\left(\frac{WS_A}{H_0}\right) + \frac{H_1}{W}\sinh\left(\frac{W(S-S_A)}{H_0}\right)\right\} \\ \left[1 + a(T_1 - T_0)\frac{1}{EA}\left[H_1\cosh\frac{WS}{2H_1} - H_0\cosh\frac{WS}{2H_0}\right]\right] \end{array} \quad (16)$$

3 Modelica implementation

Conductor and span parameters are stored in separated records named *ConductorData* and *SpanData*. Day of the year and time of the day are stored in a third record whose name is *TimeData*. Tables 1 to 3 summarize the parameters in each record.

3.1 Thermal model

14 functions have been defined for the implementation of the thermal model (see table 4). 3 classes have been also designed:

ConvectionHeatFlow: a model similar to the *HeatTransfer.Convection* model, but whose parameters are driven by the conductor, span and time parameters.

declaration	**meaning**
`Real D`	External diameter
`Real a`	Coefficient of dilatation
`Real E`	Module of elasticity
`Real W`	Linear weight
`Real A`	Cross section area
`Real C`	Linear heat capacity
`Real R_ref`	Linear electrical resistence
`Real T_ref`	Temperature of reference
`Real alpha`	Slope of resistance change
`Real abs =0.5`	Absorvity
`Real emi =1.0`	Emissivity

Table 1: Parameters in the *ConductorData* record

declaration	**meaning**
`Real He`	Altitude above sea level in m
`Real L`	Latitude in deg
`Real Zl`	Azimuth of the line in deg
`Real S`	Span length en m
`Real Dy`	Support difference of level in m
`Real T_0`	Temperature of conductor in state of reference in K
`Real Ten_0`	Tension of conductor in state of reference in KgF
`Real L_0`	Lenght of conductor in state of reference in m

Table 2: Parameters in the *SpanData* record

declaration	**meaning**
`Integer Day`	Day of the year (1-365)
`Real Hour`	Time of the day (0-24, 13.5 means 1:30 pm)

Table 3: Parameters in the *TimeData* record

SolarHeatFlow: a model similar to the *HeatTransfer.PrescribedHeatFlow* model, but whose value is driven by sun and span position.

StandAloneHeatingResistor: a model similar to the *Electrical.Analog.Basic.HeatingResistor* model, that also computes the heat gain by Joule effect for a certain electrical current.

Conductor: it is a Heat Capacitor with a temperature signal port.

3.2 Mechanical and geometrical model

Mechanical and geometrical model is implemented by 4 functions (table 5) and 4 main classes:

CatenaryStateChange: this class is the implementation of the state change equations 15 and 16. (See File 1). This class is perhaps, the most important of the mechanical model. Here we stablish that the state equation must also satisfy the new length condition of 13. In order to help a fast solution of the state change equations, a starting point for $\alpha = H/W$ can be set. We suggest to use 300 [2].

Catenary: it is a partial class that joins the thermal, mechanical and geometrical models (see 2). In one hand it implements equation 1 as a *Conductor* (i.e. a heat capacitor) whose heat port has attached models for the heat flows (joule effect q_J, solar radiation q_s, convection q_c and radiation q_r); air temperature T_a is included as a prescribed temperature. In the other hand it has also a component of the *CatenaryStateChange* class; the temperature of the Conductor is used as an input for the State Change analysys, whose main output is the sag D calculation. In order to use this class, a derived class must be designed, so the electrical current I is defined. See File 3.

ElectricalCatenary: it is a derived class from *Catenary* class, in which a electrical current signal is attached to the conductor

[2] As an example, in the simulation shown in section 4.1 the numerical methods have found $\alpha = 394.7$ for $t = 0$

function	**compute:**
`AirConductivity`	thermal air conductivity as a function of film air temperature
`AirDensity`	air density as a function of altitude and film air temperature
`AirViscosity`	air viscosity as a function of film air temperature
`AngleFactor`	correction factor for convection heat losses as a function of the angle between the wind and conductor
`Asinh`	$asinh(x)$
`ConvectionFlow`	convection heat losses
`ForcedConvectionHigh`	forced convection heat losses for high wind speeds
`ForcedConvectionLow`	forced convection heat losses for low wind speeds
`ForcedConvection`	forced convection heat losses for any wind speed
`NaturalConvection`	natural convection heat losses
`FilmTemperature`	film air temperature as a function of air and conductor temperatures
`SolarAltitude`	sun Altitude as a function of latidude, day of the year and time of the day
`SolarAzimuth`	sun azimuth as a function of latidude, day of the year and time of the day
`SolarFlux`	solar heat gain as a functin of altitude, solar altitude, sun and conductor azimuths and type of atmosphere

Table 4: Functions for the thermal model

function	**compute:**
`CatenaryLenght`	equation 13
`CatenarySag`	equation 12
`CatenarySa`	equation 8
`CatenaryXbar`	equation 3

Table 5: Functions for the mechanical and geometrical model

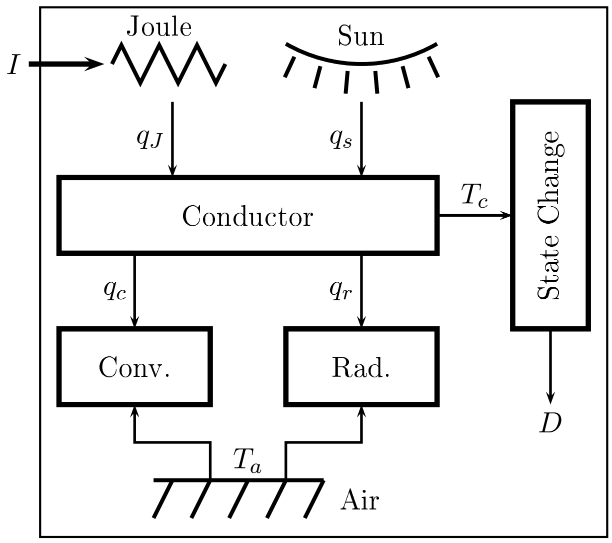

Figure 2: Catenary model

StandAloneCatenary: it is a derived class from *Catenary* class, the electrical current value is a Real, without attaching any signal to the conductor. In File 2 we show how to use this class for some specific simulation conditions.

4 System analysis and examples

In this section we illustrate the analysis capabilities of the model. To do it, we have applied the model to a real span of a 230KV distribution system in Bogotá, Colombia. The model was implemented using OpenModelica. As a part of a previous cargability study the span was modeled in detail ([5]). Most of the parameters were available (see table 6), however some experiments were conducted in order to identify the actual values of the following parameters:

- Emissivity.
- Absorvity.
- Conductor length in the state of reference.

Span parameters	
Conductor	Peacock
Altitude	2600 $msnm$
Latitude	4.779423^o $Norte$
Azimuth	76.14^o
Horizontal distance between supports	82.31 m
Difference of levels between supports	0.4; m
Nominal longitudinal tension	2970 Kgf
Conductor parameters	
External diameter	24.2 mm
Linear resistence at 25^{o}C	$9.7 * 10^{-5}$ ohm/m
Linear resistence at 75^{o}C	0.000116 ohm/m
Aluminium mass	0.79716 Kg/m
Steel mass	0.31227 Kg/m
Linear weigth	1.16 Kg/m
Nominal elasticity module	$0.7530 * 10^6$ KG/cm^2
Nominal coefficient of dilatation	$19.73 * 10^{-6}$ $1/^oC$
Cross section area	$3,4638$ cm^2

Table 6: Parameters for the examples

4.1 Heating analysis

First we study the change of conductor temperature for specific operation conditions. In this example we vary just two operation conditions through a 24 hour period: electrical current I (Figure 3) and air temperature T_a (Figure 4). These operation conditions may be considered as inputs for the present analysis where as the conductor temperature T shown in figure 5 is computed as an output.

4.2 Cargability analysis

Cargability analysis is the study of the maximum amount of electrical current that can pass through the conductor without getting a limit condition. In this example we state the limit condition as a conductor temperature of 75$^o C$ and suppose the air temperature profile shown in figure 6. Cargability has been drawn in figure 7. Notice that the electrical current now is considered as an output of the model. The simulated model is shown in File 4.

4.3 Sag analysis

We now analize the relation between Sag and three variables: electrical current I, air temperature T_a and conductor temperature T. We use the current profile of figure 3 and the air temperature profile of figure 6, and plot the sag D against the three variables (figures 8 to 10).

Notice that neither I nor T_a can explain alone the sag D. As the 24 hour cycles of I and T_a have different shapes (figures 3 and 6), and because of the nonlinear dynamic nature of the phenomena, during the 24 hour the same electrical current can occur two or more times with different sags.

Also notice that inspite the non linearities present in the equations of section 2.2, the relation between conductor temperature T and sag D is almost linear. This is why some facilities are now measuring the conductor temperature online, and not just the electrical current, in order to study the real online cargability.

4.4 Catenary analysis

We can also draw the catenary. To do that we use the dummy variable $x = St$ using $\dot{x} = S$, and use equation 3 in a simulation with stop time of 1s. An example is shown in figure 11

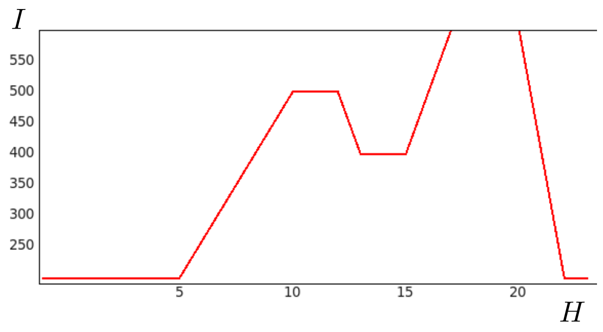

Figure 3: Example 1. Electrical current I (A) vs Time of the day H (h)

5 Conclusions

A model for bare overhead conductors has been presented. It combines thermal, mechanical and geometrical phenomena. The core of the thermal model is a heat capacitor whose parameters and

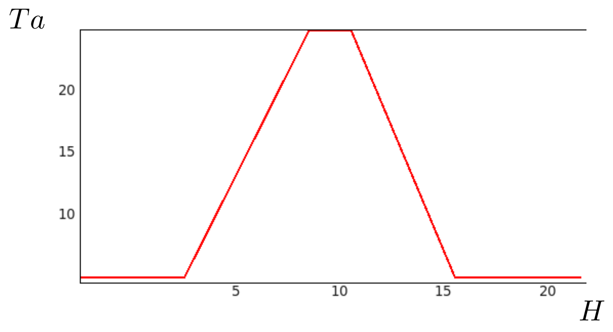

Figure 4: Example 1. Air temperature Ta (^{o}C) vs Time of the day H (h)

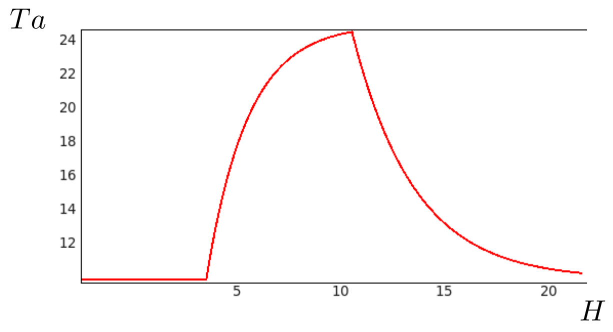

Figure 6: Example 2. Air temperature Ta (^{o}C) vs Time of the day H (h)

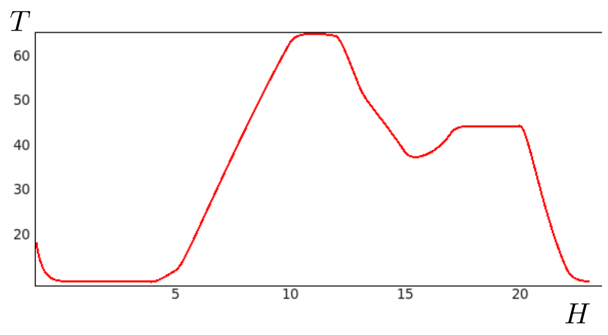

Figure 5: Example 1. Conductor temperature T (^{o}C) vs Time of the day H (h)

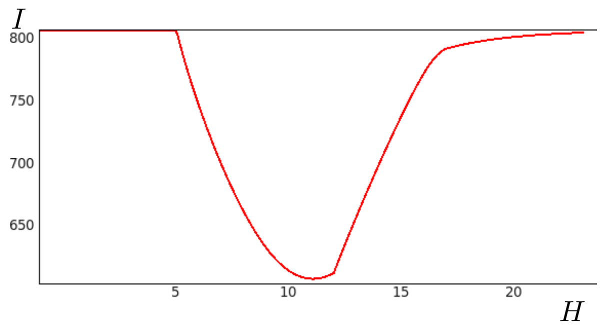

Figure 7: Example 2. Maximium electrical current I (A) vs Time of the day H (h)

inputs are driven by operation conditions as stated by the IEEE 738 standard. The mechanical model solves the state change equation for the conductor temperature of the thermal model and calculates the catenary geometry.

By using Modelica, two advantages have been found, that are clear examples of the object-oriented modeling advantages:

- The solution of the state equation is very simple. We just need to write two conditions for the conductor lenght. OpenModelica has generated the source code that we need in order to solve the equation.

- Cargability analysis is also simple. IEEE 738 standard give us a way to compute conductor temperature from an electrical current for specific operation conditions. However, the inverse problem (compute the current for a conductor temperature) is just solved for steady state in the standard. Using Modelica and OpenModelica the cargability problem is solved without any new equation.

We plan to create a Modelica library for bare overhead conductors in the short term. The library will include the parameters for the most common comercial conductors, and more analysis functionalities.

References

[1] IEEE Power Engineering Society, IEEE Standard for Calculating the Current-Temperature of Bare Overhead Conductors IEEE Std 738-2006. January 2007.

[2] OpenModelica, http://www.openmodelica.org/ last visit: november 22, 2010.

[3] Peter Fritzson, Peter Aronsson, Adrian Pop, Håkan Lundvall, Kaj Nyström, Levon Saldamli, David Broman, Anders Sandholm: OpenModelica - A Free Open-Source Environment for System Modeling, Simulation, and Teaching, IEEE International Symposium on Computer-Aided Control Systems Design, October 4-6, 2006, Munich, Germany

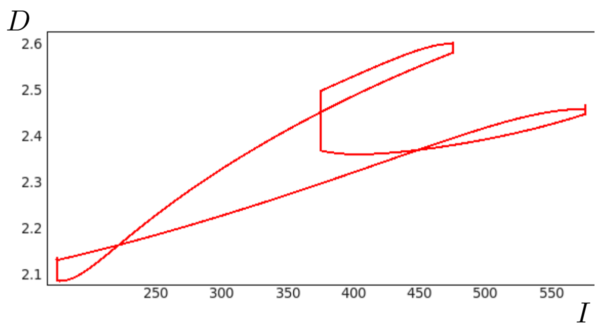

Figure 8: Example 3. Sag D (m) vs Electrical current I (A)

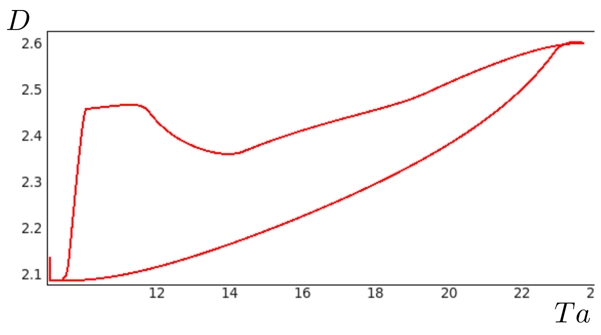

Figure 9: Example 3. Sag D (m) vs Air temperature T (°C)

[4] Donald G. Fink, H. Wayne Beaty. Standard Handbook for Electrical Engineers. McGraw-Hill Professional

[5] Oscar Duarte, Jaime Alemán, René Soto, Estrella Parra, Francisco Amórtegui, Wilson Aldana. Identificación de parámetros y estudio de cargabilidad en algunas Líneas de Transmisión de Codensa. Technical report. 2009.

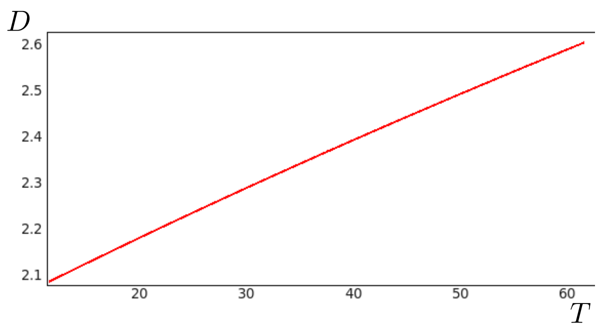

Figure 10: Example 3. Sag D (m) vs Conductor temperature T (°C)

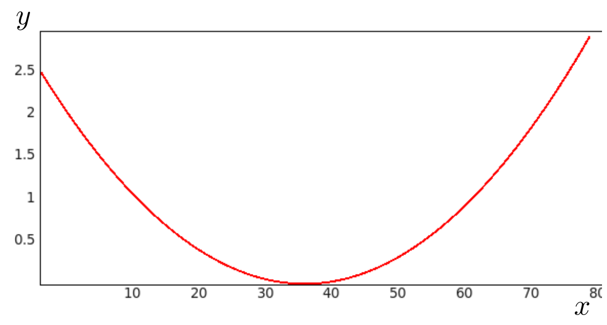

Figure 11: Example 4. Shape of the catenary. Height y (m) vs Horizontal distance x (m)

File 1: CatenaryStateChange.mo

```
within Catenary;

model CatenaryStateChange
  parameter Real W;
  parameter Real S;
  parameter Real Dy;
  parameter Real T_0;
  parameter Real Ten_0;
  parameter Real L_0;
  parameter Real a;
  parameter Real E;
  parameter Real A;
  Real T,Ten,L,H,alpha(start=300),Sag;
  TemperatureSignalPort port_T;
equation
  T=port_T.T;
  alpha=H/W;
  Ten=H*cosh(S/2/alpha);
  L=L_0*(1 + a*(T-T_0) + (Ten-Ten_0)/(E*A)
      );
  L=CatenaryLenght(S,alpha,Dy);
  Sag=CatenarySag(S,alpha,Dy);
end CatenaryStateChange;
```

File 2: MyStandAloneLine.mo

```
within Catenary;

model MyStandAloneLine
  parameter Real Imax=500;
  parameter Real Imin=200;
  parameter Real Imin2=400;
  parameter Real Imax2=600;
  parameter Real currentTable[:,2]={
     {0,Imin},
     {60*60*6,Imin},
     {60*60*11,Imax},
     {60*60*13,Imax},
     {60*60*14,Imin2},
     {60*60*16,Imin2},
     {60*60*18,Imax2},
     {60*60*21,Imax2},
     {60*60*23,Imin},
     {60*60*24,Imin}};
  parameter Real Tmax=25;
  parameter Real Tmin=5;
  parameter Real airtempTable[:,2]={
     {0,273.15+Tmin},
     {60*60*5,273.15+Tmin},
     {60*60*11,273.15+Tmax},
     {60*60*13,273.15+Tmax},
     {60*60*18,273.15+Tmin},
     {60*60*24,273.15+Tmin}};
  extends StandAloneCatenary(I(start=Imin)
      );
  Modelica.Blocks.Sources.TimeTable
      current_source(table=currentTable);
  Modelica.Blocks.Sources.Constant
      windVel(k=0.61);
  Modelica.Blocks.Sources.Constant
      windDir(k=0);
//   Modelica.Blocks.Sources.Constant
     airTemp(k=273.15+20);
  Modelica.Blocks.Sources.TimeTable
      airTemp(table=airtempTable);
  Modelica.Blocks.Sources.BooleanConstant
       atmos(k=true);
equation
  I=current_source.y;
  windVelocity=windVel.y;
  windDirection=windDir.y;
  atm=atmos.y;
  ta=airTemp.y;
end MyStandAloneLine;
```

File 3: Catenary.mo

```
within Catenary;

partial class Catenary
  parameter ConductorData con(D=24.2,a
      =19.7e-6,E=0.753e6,A=3.4638,W=1.16,C
      =909.9,R_ref=0.000097,T_ref
      =273.15+25,alpha=3.8e-7,abs=0.5,emi
      =1.0);
  parameter SpanData span(He=2600,L
      =4.779420,Zl=76.14,S=82.31,Dy=0.4,T_0
      =273.15+20,Ten_0=2970,L_0=82.54);
  parameter TimeData to(Hour=0,Day=57);
  parameter Real InitialTemp=20;
//Weather
  Real ta,taCelcius,windVelocity,
      windDirection;
  Boolean atm;
// Thermal
  Conductor Wire(C=con.C);
  Modelica.Thermal.HeatTransfer.Sources.
      PrescribedTemperature Env;
  Modelica.Thermal.HeatTransfer.Components
      .BodyRadiation Rad(Gr=con.emi*con.D
      *0.0178/5.6704);
  SolarHeatFlow Sun(He=span.He,L=span.L,
      absorvity=con.abs,Zl=span.Zl,area=con
      .D/1000,Hour=to.Hour,Day=to.Day);
  ConvectionHeatFlow Conv(axis=true,D=con.
      D,He=span.He);
//Mechanical & geometric
  CatenaryStateChange Sag(a=con.a,E=con.E,
      A=con.A,W=con.W,S=span.S,Dy=span.Dy,
      T_0=span.T_0,Ten_0=span.Ten_0,L_0=
      span.L_0);
  Real T(start=InitialTemp,fixed=false),Qj
      ,Qs,Qr,Qc,hour,sag;
equation
  hour=time/(60*60);
  T=Wire.T-273.15;
  Qj=-HR.heatPort.Q_flow;
  Qc=Conv.Q_flow;
  Qs=Sun.Q_flow;
  Qr=Rad.Q_flow;
  sag=Sag.Sag;
// weather signals to components
  Conv.Vw=windVelocity;
  Conv.phi=windDirection;
  Sun.Atm=atm;
  Env.T=ta;
  taCelcius=ta-273.15;
//Thermal
  connect(HR.heatPort,Wire.port);
  connect(Wire.port,Sun.port);
  connect(Wire.port,Rad.port_a);
  connect(Rad.port_b,Env.port);
  connect(Wire.port,Conv.solid);
  connect(Conv.fluid,Env.port);
//Mechanical and geometric
  connect(Wire.port_T,Sag.port_T);
end Catenary;
```

File 4: Cargability.mo

```
within Catenary;

model Cargability
  parameter Real TAmin=10;
  parameter Real TAmax=25;
  parameter Real TCmax=75;
  parameter Real Vvto=0.61;
  parameter Integer TAFlag=3;
  extends StandAloneCatenary;/*(
          taCelcius(start=TAmin),
          ta(start=273.15+TAmin),
          Wire.T(start=273.15+TCmax));*/
  Modelica.Blocks.Sources.Constant
      temperature_source(k=273.15+TCmax);
                                        // Ex. of
       Temparature of conductor known
  Modelica.Blocks.Sources.Constant
      windVel(k=Vvto);
  Modelica.Blocks.Sources.Constant
      windDir(k=0);
  Modelica.Blocks.Sources.Trapezoid
              airTemp1(offset=273.15+TAmin,
      amplitude=TAmax-TAmin,startTime
      =60*60*6,rising=60*60*5,width
      =60*60*2,falling=60*60*5,period
      =60*60*24);
  Modelica.Blocks.Sources.Sine
                  airTemp2(offset=273.15+(
      TAmax+TAmin)/2,amplitude=(TAmax-TAmin
      )/2,freqHz=1/(60*60*24),phase=-
      Modelica.Constants.pi/2);
  Modelica.Blocks.Sources.Exponentials
            airTemp3(offset=273.15+TAmin,
      outMax=TAmax-TAmin,startTime=60*60*6,
      riseTime=60*60*7,riseTimeConst
      =60*60*2,fallTimeConst=60*60*3);
  Modelica.Blocks.Sources.BooleanConstant
       atmos(k=true);

equation
  Wire.T=temperature_source.y;
  windVelocity=windVel.y;
  windDirection=windDir.y;
  atm=atmos.y;
  if TAFlag==1 then
    ta = airTemp1.y;
  elseif TAFlag==2 then
    ta = airTemp2.y;
  elseif TAFlag==3 then
    ta = airTemp3.y;
  end if;
end Cargability;
```

File 5: SagAnalysis.mo

```
within Catenary;

model SagAnalysis
  parameter Real TAmin=10;
  parameter Real TAmax=25;
  parameter Real Imax=500;
  parameter Real Imin=200;
  parameter Real Imin2=400;
  parameter Real Imax2=600;
  parameter Real mitable[:,2]={
      {0,Imin},
      {60*60*6,Imin},
      {60*60*11,Imax},
      {60*60*13,Imax},
      {60*60*14,Imin2},
      {60*60*16,Imin2},
      {60*60*18,Imax2},
      {60*60*21,Imax2},
      {60*60*23,Imin},
      {60*60*24,Imin}};
  extends StandAloneCatenary(T(start=60));
  Modelica.Blocks.Sources.TimeTable
      current_source(table=mitable);
  Modelica.Blocks.Sources.Constant
      windVel(k=0.61);
  Modelica.Blocks.Sources.Constant
      windDir(k=0);
//  Modelica.Blocks.Sources.Constant
    airTemp(k=273.15+20);
  Modelica.Blocks.Sources.Exponentials
            airTemp(offset=273.15+TAmin,
      outMax=TAmax-TAmin,startTime=60*60*6,
      riseTime=60*60*7,riseTimeConst
      =60*60*2,fallTimeConst=60*60*3);
  Modelica.Blocks.Sources.BooleanConstant
       atmos(k=true);
equation
  I=current_source.y;
  windVelocity=windVel.y;
  windDirection=windDir.y;
  atm=atmos.y;
  ta=airTemp.y;
end SagAnalysis;
```