Norwegian University
of Life Sciences

Master's Thesis 2017 30 ECTS
Faculty of Science and Technology

Electromagnetic modelling of power umbilical systems

Martin Hovde

Preface

The topic for this thesis was proposed by Dr. Marius Hatlo at the Technical Analysis Center at Nexans' facility in Halden. A computer program where a user can model and simulate power umbilicals with arbitrary terminations at each end were desired. The computer program should be able to calculate currents and voltages on each conducting element inside the power umbilical, at any point along its length. Another important aspect were that a user should be able to conduct harmonic analyses on power umbilical systems. Even though it is still in a developmental stage, the fundamental parts of such a program has been established.

This thesis marks an end to my time as a student at the Norwegian University of Life Sciences, where I for the last six years have enjoyed studying physics and applied mathematics. The specialization courses I have taken have always been determined by my fields of interest. I could therefore not have been more pleased with the topic for my master's thesis, which turned out to be a multidisciplinary task involving electromagnetics, electrical engineering and computer programming.

There are many people that deserve a thought with regards to this thesis. First and foremost, I want to thank Dr. Marius Hatlo for dedicated and skillful supervising throughout the semester. It is not every student writing a thesis for the private sector that is lucky enough to have a supervisor with a strong background in physics, but I am certainly happy that I was. I also want to thank Assoc. Prof. Arne Auen Grimenes for proficient guidance and opportunities granted. In addition, I want to thank Prof. Bjørn Fredrik Nielsen for advice and for being available for questions.

I also want to thank all the outstanding people I have met in Ås for enjoyable years. A special thanks go to my parents, for always being there for me and supporting me.

Sarpsborg, June 15th 2017

Martin Hovde

Sammendrag

I strømførende kabler med flere elektriske ledende materialer vil disse lederne påvirke hverandre gjennom elektromagnetiske felt. Dette kan føre til at ledere som i utgangspunktet ikke er tiltenkt å være spenningssatt kan få induserte spenninger og strømmer.

Nexans, som er en verdensledende aktør innen produksjon av kraftkabler, produserer det som kalles kraft-navlestrengskabler. Disse kraft-navlestrengskabelene består av mange typer ledende elementer, slik som kraftfaser (høyspent), stålrør, elektriske elementer (lavspent) og armering rundt fiberoptiske elementer. I tillegg ligger det beskyttende stålarmering snodd rundt hele navlestrengskabelen.

Grunnet dette store antallet ledende elementer som ligger svært tett pakket, der noen ledere fører store strømmer ved høye spenninger, er det viktig å kunne forutsi nivået på eventuelle spenninger og strømmer som blir indusert. En kompliserende faktor er også at kraftelektronikk, eller andre ulineære komponenter plassert før eller etter kraft-navlestrengskabelen, kan føre til at overharmoniske spenninger og strømmer introduseres i kraftsystemet. Problemstillingen for denne masteroppgaven, gjennomført ved Teknisk Analysesenter ved Nexans i Halden, er derfor å utvikle et dataprogram der en bruker kan beregne strømmer og spenninger i et hvert ledende element i en vilkårlig kraft-navlestrengskabel, med mulighet for å utføre overharmonisk analyse.

Den bakenforliggende modellen som presenteres i denne oppgaven består av to deler; den første bestanddelen er den analytiske løsningen til telegraflikningene for flerledere; den andre bestanddelen er en klassisk formulering av impedans- og admittansmatriser for kabler.

Det tilhørende dataprogrammet er skrevet i Python. Slik det er idag kan en bruker modellere en kraft-navlestrengskabel med brukerspesifiserte kraftfaser, metallrør og omgivelser, og med vilkårlige termineringer i hver ende.

Dataprogrammet er validert ved sammenlikning med analytiske beregninger, simuleringer i elementmetode-programmet Flux2D og med målinger utført på en navlestrengskabel.

De to første valideringsmetodene er i svært god overensstemmelse med programpakken, selv om noen avvik observeres mot Flux2D, grunnet at modellen fremlagt i oppgaven ikke modellerer nærhetseffekter. Det er ikke implementert noen fysisk riktig formulering av armeringen til kraft-navlestrengskabelene, noe som er utslagsgivende i sammenlikninger med målinger. Parametertilpasning øker her overensstemmelsen.

Abstract

In current carrying cables with multiple electrical conductors, there will be interactions between the conductors due to electromagnetic fields. Not all of these conductors are excited by an external voltage source, but unintended voltages and currents can arise through electromagnetic induction.

Nexans, as one of the worlds leading cable manufacturers, produce what is called power umbilicals. These power umbilicals contain a variety of different conductors, such as high-voltage power phases, steel tubes, low-voltage electrical elements and armour surrounding fiber optic elements. In addition, a steel armour is placed around the power umbilical for mechanical protection.

Due to a large number of tightly packed conductors, where some carries substantial currents at high voltage, it is crucial to be able to predict the magnitude of the voltage and currents that may be induced in the different conductors. A complicating matter is that the existence of power electronics placed at one or both ends of the power umbilical, can heavily distort the voltages and currents due to injection of harmonic content. The problem at hand for this thesis, conducted at the Technical Analysis Center at Nexans in Halden, is therefore to develop a computer program where a user can calculate currents and voltages in every conducting element inside an arbitrary power umbilical. Additionally, a user should be able to conduct harmonic analysis.

The underlying model of the computer program can be viewed as having two main parts; The first constituent is the analytical solution to the multiconductor transmission line telegrapher's equations. The second constituent is a classic formulation of the impedance- and admittance matrices for cables.

The accompanying computer program is written in Python. As of now, a user can model a power umbilical with user specified power phases, metal tubes and surroundings, and with arbitrary terminations at each end.

The computer program and the model is validated against analytical methods, simulations in the finite element software Flux2D, and lastly, against comparisons with measurements conducted on a power umbilical.

The two former methods are in good agreement with the simulations in the computer program, even though some deviations are observed in comparison to Flux2D, due to the proposed model not being aware of proximity effects. No formulation for the armour of power umbilicals is implemented, which is prominent when comparing simulations with measurements. Best fit parameters are found to yield lower deviations.

Contents

1. Introduction

With the most accessible offshore oil and gas resources being depleted, the offshore industry is forced further from shore and into deeper waters. As the reservoirs become increasingly more difficult to reach, the requirements for the technology used in production facilities are gradually becoming more stringent. An important part of every offshore oil and gas production facility is the power system, both topside and subsea.

The topside system, if located offshore, generates power by the use of gas turbine-driven synchronous generators. These generators supplies power to loads on the platform, as well as to the subsea power system. The power generated is transmitted through step-out cables to the subsea distribution system, where switchgear on the seabed connects and distributes power to various loads. The tendency in new facilities is to have increasingly longer step-out cable lengths and more of the equipment installed subsea, rather than topside.

It is not only electric power that needs to be delivered to the subsea system. Signal transfer and communication with equipment on the seabed is also needed, as well as transportation of fluids, e.g. for injection into the production stream in order to optimize production.

As of now a smaller, but still relevant application, is related to wind turbines in offshore wind farms. Aside from needing power cables to transmit the produced power to the utility grid, the wind turbines also need communication and sensory equipment for controlling the turbines rotor speed, blade angle, temperature, hydraulics and so on.

In both cases it will generally be more expensive to install numerous, several kilometers long cables and/or pipes. A motivation for a single cable which provides all the needed consumables is therefore present.

For Nexans, which is one of the world's largest cable manufacturers, the concept of *umbilical cables* emerged in the early nineties, with the first delivery being a steel tube umbilical in 1993. Since then over 1800 kilometers of umbilical cable has been supplied by Nexans, with a variety of different designs and intended applications.

The name "umbilical" originates from the umbilical cord of embryos and fetuses, but in the engineering sense, an umbilical is simply a cable or hose supplying consumables of more than one type to a given load, e.g. a hose supplying a diver with air and communication is considered an umbilical.

Nexans generally divides their umbilicals into two groups - control- and power umbilicals. The control umbilicals usually contain low-voltage electrical elements for

control systems, steel tubes for transportation of fluids and/or fiber optical cables used for communication and distributed temperature sensing (DTS). Power umbilicals may contain all of the elements in a control umbilical, but in addition it contains high-voltage power cables.

The power umbilicals produced by Nexans vary in size, number of elements and design. It may or may not contain steel tubes, it may contain 3, 6 or 9 power phases, which may be screened or unscreened, with the power phases twisted either in the same or different layers, et cetera. Although these variations may seem small and insignificant to people outside the cable industry, they can heavily impact the mechanical, electrical and thermal properties of the cables.

A general computational tool in which a power umbilical can be somewhat quickly specified and analyzed is desired by Nexans. The goal for this thesis is therefore to establish a general electromagnetic model for power umbilical systems implemented in a computer program.

1.1 Scope of work

The main goal for this thesis will be to develop an electromagnetic model for power umbilical systems. The model will be implemented in a accompanying computer program package written in Python. With the computer program, a user will for example be able to calculate currents and voltages in each conducting element in a power umbilical at any point along its length.

The model will be general and applicable to numerous power umbilicals, and as a mean of verification a few worked examples will be presented.

The model will be based on what is known as the *telegrapher's equations* for multiconductor transmission lines, which is a set of linear first-order partial differential equations used for describing long transmission lines and power cables. An analytical solution to these equations can be found. This means that one can solve a set of algebraic equations, instead of numerically solving a set of differential equations. The difficult task, however, is to determine the electrical parameters for the cables.

Inspired by Gustavsen et al. (2009), the program package will contain a library with different types of elements that are commonly used in power umbilicals. The user can choose from the library which elements to include, specifying parameters related to electrical properties, placement, size of conductors, and so on. The user may also add new elements to the library if needed. The computer program will output different data and plots that are of relevance to the user. The computer program will also allow a user to input predetermined electrical parameters.

An important part of the computer program is that a user should be able to conduct *harmonic analyses* on power umbilical systems, where the response of the power umbilical to a known harmonic spectrum can be found.

2. Theory

2.1 Subsea power systems

Offshore oil- and gas reservoirs can be located several kilometres away from the topside processing- and storage units. Thus, the extraction of gas- or crude oil requires transportation over long distances through pipelines. To boost the rate of production, enhance recovery and to cope with pressure drops along these pipelines, compressors and pumps driven by electrical motors may be installed on the seabed to elevate pressure levels. These pumps and compressors are parts of what is referred to as the *subsea processing units*.

As with any power system, the three main parts of the subsea power system is the source, the distribution equipment and the load.

The source of electrical power for the subsea power system is the topside power system. This can either be an autonomous power system placed on platforms or the on-shore utility grid. If located on platforms, the power is produced by large gas-turbine driven synchronous generators.

Depending on cable length and network topology, the source for the subsea power system has a nominal voltage ranging from a few kV up to 145 kV. The upper voltage limit arises due to the use of dry-mate connectors used in the termination of cables. If wet-mate connectors are used, the upper voltage limit is 36 kV. The fundamental frequency is either 50 or 60 Hz.

The connection between the topside power system and the subsea power system is the step-out cable. The step-out cables can be up to 200 kilometres long, being either a more traditional three-core power cable or a power umbilical. Depending on the length of the cable, there may be installed step-up and step-down transformers on the sending and receiving end of the cable, respectively. Since power umbilicals used as step-out cables is the main focus of this thesis, the details in the design of power cables and umbilicals will be dealt with in detail in the upcoming sections.

The electrical motors driving the subsea processing units are the main load of the subsea power system, with nominal voltages of ≤ 6.6 kV. The mechanical power rating of the gas compressors can be several megawatts, such as the 12.5 MW gas compressor used in Ormen Lange or the two 11.5 MW compressors used in the Åsgård Subsea Gas Compression system. Condensate pumps have mechanical power ratings up to a few megawatts. To operate these motors in a flexible and controlled manner, variable speed drives (VSDs) are used. VSDs are usually placed topside, but in the future the trend will be to place them subsea (Garvik 2015). The operating voltages and power ratings of the VSDs must match the ratings of the equipment

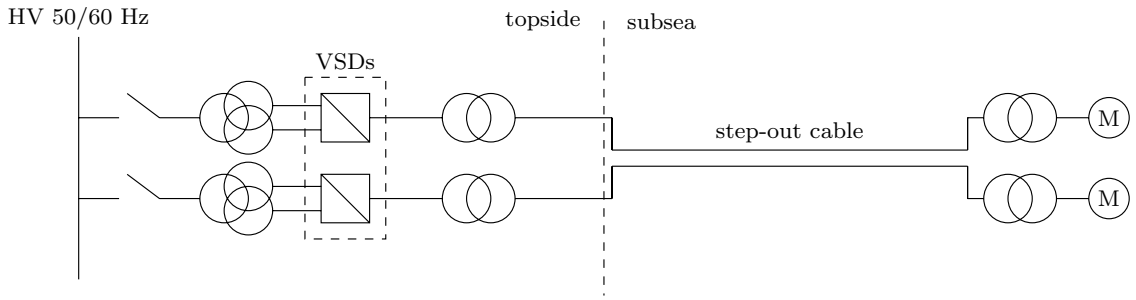


Figure 2.1: Example of a subsea power system with topside variable speed drives and a long step-out cable delivering electrical power to motors on the seabed.

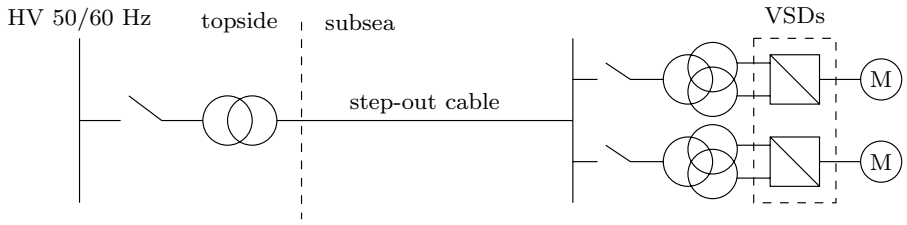


Figure 2.2: Example of a future subsea power system with subsea variable speed drives and a long step-out cable delivering electrical power to motors on the seabed.

connected to it.

As a mean of power factor correction and voltage regulation, static var compensators (SVCs) may be installed.

In Fig. 2.1 a typical subsea network topology is depicted in a single-line diagram. This topology has topside VSDs and a long step-out cable. On the far left, the subsea power system is fed by the topside system with high voltage and a fundamental frequency of either 50 or 60 Hz. To reduce the amount of harmonics in the topside power system, multi-winding transformers are connected to the VSDs, which again connects to a step-up transformer. The power is then transferred through a long step-out cable. On the seabed, the step-down transformers feed the electrical motors connected to subsea processing units. Note that there are two three-phase circuits in the step-out cable of Fig. 2.1. This could, for example, be the inner and outer circuit of the power umbilical depicted in Fig. 2.5.

Fig. 2.1 represents a traditional subsea power system, while Fig. 2.2 shows an example of what topologies will probably look like in future subsea power systems. Here, the VSDs are placed subsea. Note that both the traditional and modern topology can have short step-out cables instead, where step-down and step-up transformers can be left-out for economical reasons.

The existence of power electronics in subsea power systems cause injection of *harmonic content*. In Chapter 5 the effect harmonic content may have on power umbilicals will be studied.

2.1.1 Power cables

Modern cross-linked polyethylene (XLPE) insulated cables have been around for the last 50 years or so. They are usually designed in approximately the same manner, with the same type of elements. In Fig. 2.3 a typical single-core (SC) power cable is depicted. A three-phase supply may consist of three separate SC cables, or it may be in the form of what is called a three-core cable. A three-core cable can either be surrounded by an armour comprised of twisted steel wires, or a solid metallic tube. The latter is called a *pipe-type* (PT) cable. A PT cable can be seen in Fig. 2.4. For the upcoming description of a power cable, the reader can use Fig. 2.3 as reference.

The *conductor* carries the electrical current. In power cables it is usually made of either aluminum or copper. The material used depends on several factors such as conductivity, cost, weight, tensile strength etc. Even though the conductor depicted in Fig. 2.3 is solid, conductors may also be stranded. Stranded conductors provide a mean of dealing with unwanted electromagnetic effects, and is typically used in cables with a larger cross-sectional area.

Surrounding the conductor is a thin *semi-conducting layer*. The purpose of this semi-conducting layer is to smooth out any irregularities on the surface of the conductor. This is necessary to avoid any spikes in the electric field strength around any bumps or voids, and to avoid partial discharges over small gaps. The semi-conducting material is similar to the insulating material, but is heavily doped with carbon to make it conducting.

The *insulation* isolates the conductor from any other conducting material. The insulation must be able to withstand the SC cable's electrical field under both under normal operation nor during transient voltage spikes. Traditionally the insulation of power cables consisted of oil-impregnated sheets of paper. While cables with this type of insulation is still produced, extruded XLPE insulation is more common. XLPE is considered to be more environmentally friendly, and it also has the advantage that it can withstand temperature up to 90 °C, compared to 60 °C for paper-insulation.

Surrounding the insulation is another layer of semi-conductive material, with the outer layer serving the same purpose as the inner.

To minimize the electromagnetic fields surrounding the cable, thus preventing *crosstalk* with other equipment and conductors, a metallic *screen* is often used in high-voltage power cables. The screen can be made of different metals, and may be stranded or massive.

Armour and *sheaths* function as mechanical stiffeners and protects the vital parts of the cable from the surrounding environment. They also serve as insulation for the metallic screen.

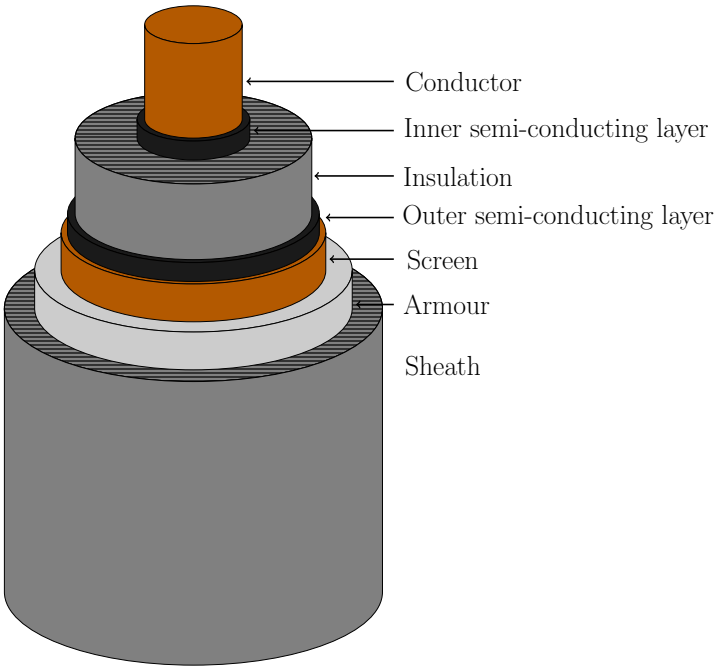


Figure 2.3: A single-core XLPE insulated power cable.

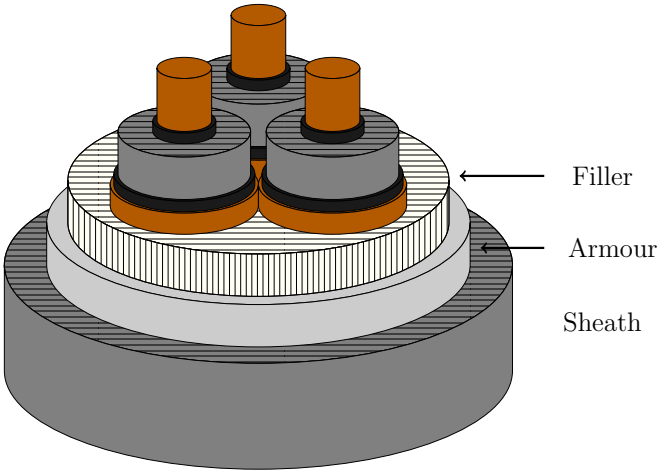


Figure 2.4: A pipe-type power cable consisting of three single-core cables.

2.1.2 Power umbilicals

A power umbilical is in many situations used as a step-out cable in subsea power systems, with one of the reasons being that it is economical to decrease the number of pipelines and/or cables connecting the topside system to the subsea system.

An example of a power umbilical produced by Nexans is shown in Fig. 2.5. The power to the subsea processing units are delivered through the *power phases*, which are essentially SC cables, and hence their design is similar to that of power cables explained in the previous section. The power phases may have a metallic screen, even though this is not the case of the power phases in Fig. 2.5. The example umbilical consists of two three-phase arrangements, naturally referred to as the inner and

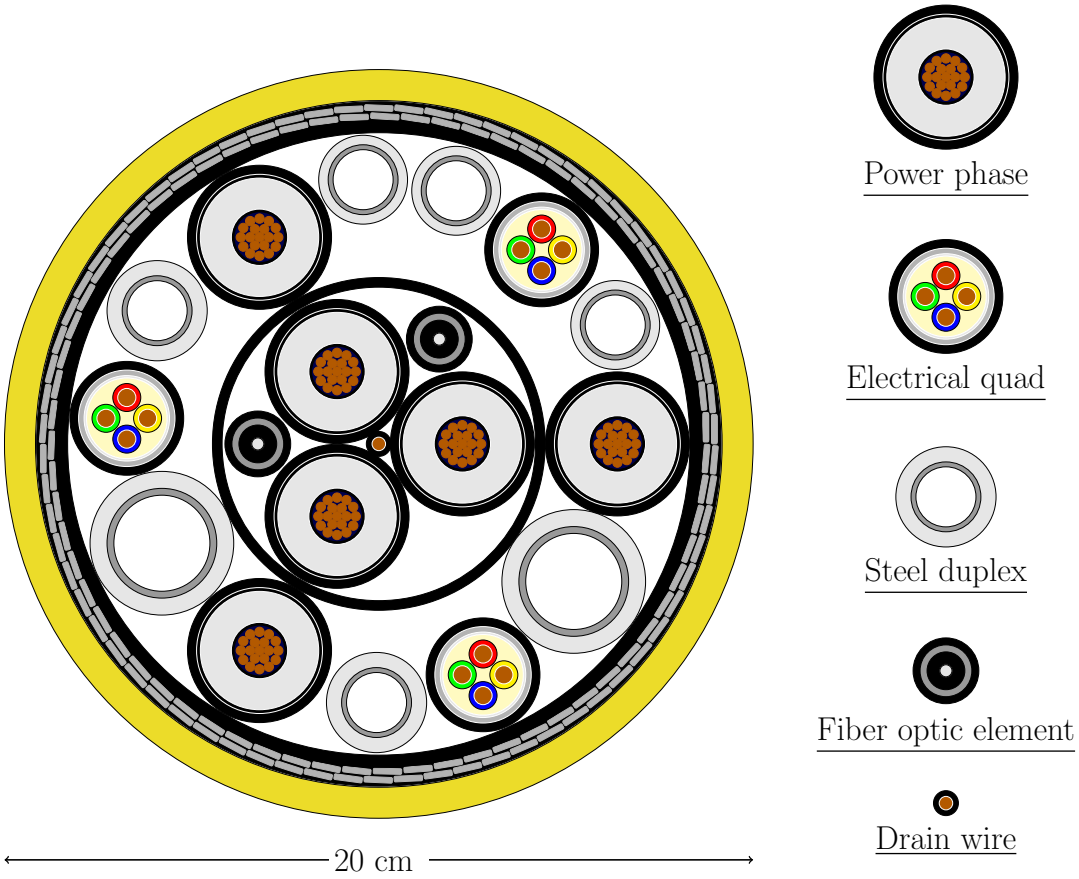


Figure 2.5: A power umbilical produced by Nexans.

outer circuit.

The umbilical also have several *electrical quad* cables. These cables are generally used for control applications, and normally have a voltage rating up to a few kV. Each electrical quad is either operated as a two- or four-wire transmission line, depending on the application. Due to a much lower voltage rating, the design of the electrical quads differ from power cables. General "guidelines" in design still apply, such as semi-conducting layers to ensure a radially uniform electric field, or the use of some kind of dielectric as insulating material.

There are *steel duplex tubes* in the umbilical that serve various purposes, e.g. carrying fluids that are injected into the production stream in order to optimize production. They are made of super-duplex steel and are fitted with a high-density polyethylene (HDPE) sheath.

A small *drain wire* can be seen at the center of the cable, providing a low-resistance path for common mode currents. Two *fiber optic elements* are also present in the inner twisting layer, which are used for signal and communication purposes. The fiber optic elements have their own armour, to increase mechanical strength and prevent breakage.

The outer grey stranded ring enclosing all cable elements is the *armour*, which functions as a mechanical stiffener. The armour can be made of a variety of different steels. Surrounding the armour is a HDPE sheath.

Although not shown in Fig. 2.5, polyethylene filler elements are placed around

the elements inside the power umbilical, to make sure that all elements stay put. Often, these are hollow.

In addition to the elements in the power umbilical, there are a few things that should be noted in the implementation of power umbilicals as step-out cables. Firstly, during operation, the areas surrounding the elements in Fig. 2.5 and inside the hollow filler elements (not shown) are flooded by seawater. This is to avoid the cable being crushed by the immense pressure at large depths. Secondly, all non-excited elements in the umbilicals such as the steel duplex tubes or armour are either grounded in each end or left open. More often than not they are grounded, to avoid any voltages from arising in these elements.

Which elements included in Nexans' power umbilicals vary, and Fig. 2.5 is only one specific example.

2.2 General method for electromagnetic modelling of transmission lines

Most of the theory that will be presented in this section can be found in Paul (2008).

2.2.1 Electrical parameters

Any given cross section of a conductive transmission line (TL) is defined by four electrical parameters known as the *primary line constants*. These parameters are *resistance, inductance, conductance* and *capacitance*, and are usually denoted r, l, g and c respectively. Whenever these parameters are mentioned with lower-case symbols in this thesis, they are referred to as *per unit length*, unless otherwise stated.

For a two-conductor transmission line, these parameters are scalars. For TLs or cables with $n+1$ conductors, the conductor parameters are given in matrices with dimensions $n \times n$, as will be shown in section 2.2.2.

Resistance

The resistance of a conductive material will depend on whether direct or alternating current is running through it. The per unit length dc resistance of a uniform conductor with conductivity σ and cross-sectional area A is given as

$$r_{dc} = \frac{1}{\sigma A} \quad [\Omega/\mathrm{m}] \tag{2.1}$$

The resistance of metals vary with temperature. For adequately small temperature variations a linear dependence is assumed, and the dc resistance at temperature T in Celsius is found as

$$r_{dc}(T) = r_{dc,\ 20\ ^\circ\mathrm{C}}(1 + \alpha_T(T - 20^\circ\mathrm{C})) \quad [\Omega/\mathrm{m}] \tag{2.2}$$

where $r_{dc,\ 20\ ^\circ\mathrm{C}}$ is the conductor resistance at 20 °C and α_T is the *temperature coefficient of resistance* with units $^\circ\mathrm{C}^{-1}$.

A uniform conductor carrying a dc current will have a uniform current distribution, but this is not the case for uniform conductors carrying ac currents. Instead the current distribution will become increasingly more nonuniform as the electrical frequency increases, with current migrating towards the surface of the conductor. This phenomenon is known as the *skin effect*. The net result of the skin effect on resistance is that the effective current-carrying cross-sectional area of the conductor decreases, and hence the resistance increases according to Eq. (2.1). A more in-depth explanation of skin effect is given in section 2.3.2.

Another effect that causes the resistance of conductors carrying ac currents to increase is the *proximity effect*, which is prominent for closely packed conductors. Due to electromagnetic effects, the current-density inside two or more conductors in close proximity will be nonuniform, which will yield an increase in resistance due to a decrease in the effective current-carrying area.

There are several ways to model these effects, ranging from analytical to empirical methods. In this thesis, however, modelling of proximity effects will be omitted.

Conductance

Conductance of TLs exist both between conductors and between conductors and ground. Conductance accounts for current leakage through dielectrics, and is a measure of how easily conduction current can pass through it, i.e. conductance is the inverse of resistance.

Ohm's law for transverse current flow is therefore given as

$$g = \frac{I_t}{V} \quad [\mathrm{S/m}] \tag{2.3}$$

where I_t is the per unit length transverse conduction current flowing through the dielectric and V is the electric potential difference between two conductors or two conductors and ground. More formal definitions in terms of electromagnetic field theory are available in relevant literature (Paul 2008).

Inductance

Consider a closed circuit consisting of a conductor carrying a current I and its return-conductor. The open surface S enclosed by the circuit will be penetrated by a magnetic field $\mathbf{B}$, called the *magnetic flux density*, which is created by the flowing currents. The magnetic flux Ψ that penetrates S is given by

$$\Psi = \iint_S \mathbf{B} \cdot d\mathbf{S}$$

If the material surrounding the conductors is *linear*, then there exists a linear relationship between Ψ and I, with the constant of proportionality L given as

$$L = \frac{\Psi}{I} \quad [\mathrm{H}]$$

which is the inductance of the circuit given in henrys. The per unit length inductance l can then be found by finding the resulting magnetic flux from a segment with length

Δz as

$$l = \frac{\Psi}{I\Delta z} = \frac{\psi}{I} \quad [\text{H/m}] \tag{2.4}$$

where ψ is the per unit length magnetic flux.

As explained above the magnetic flux lines can cut the surface enclosed by the circuit itself, but they can also cut through surfaces enclosed by other, neighbouring circuits. The former is related to *self-inductance* and the latter to *mutual-inductance*. The self-inductance of a circuit i is

$$l_i = \frac{\psi_i}{I_i} \tag{2.5}$$

where ψ_i is the per unit length magnetic flux through the circuit loop due to current I_i flowing (in the same loop). If there exists another neighbouring circuit j with current I_j, the mutual-inductance of i due to j is

$$l_{ij} = \frac{\psi_i}{I_j}|_{I_i=0} \tag{2.6}$$

where ψ_i is the magnetic flux through circuit i due to current I_j when I_i is set to zero.

Capacitance

Capacitance of a transmission line is due to differences in electric potentials between conductors, or between conductors and ground. The per unit length capacitance of a conductor holding a charge q per unit length at voltage V, is given in farads per meter as

$$c = \frac{q}{V} \quad [\text{F/m}] \tag{2.7}$$

where q is in units coulombs per meter. The above equation can then be used to determine the capacitance between two conductors or between a conductor and ground. The former is obtained if V is taken as the voltage between two conductors at interest and the latter if V is taken as the voltage-to-neutral.

Transmission lines excited with an ac voltage will cause the conductors to have varying electrical potentials, and therefore varying charge distribution. This sinusoidally varying charging and discharing of the conductors is called the *charging current*, and will flow even if the transmission line is open-circuited. This is due to capacitance being a shunt admittance, just as conductance. The difference is that the capacitance admits a *displacement current* rather than a conduction current.

2.2.2 Distributed parameters model and the telegrapher's equations

For transmission lines longer than a certain length, the classical lumped parameter circuit model do not yield sufficiently accurate results. In reality, the circuit elements are distributed continuously along the transmission lines. Distributed parameters models will first be derived for a two-conductor transmission line and then for a general multiconductor transmission line (MTL) with $n+1$ conductors.

Two-conductor transmission line

Consider the circuit shown in Fig. 2.6. The circuit represents a small segment of a transmission line with length Δz and per unit length parameters r, l, g and c. Applying Kirchhoff's voltage law (KVL) on the outer loop of this circuit yields

$$V(z,t) - r\Delta z I(z,t) - l\Delta z \frac{\partial I(z,t)}{\partial t} - V(z+\Delta z,t) = 0$$

or

$$-\frac{V(z+\Delta z,t) - V(z,t)}{\Delta z} = rI(z,t) + l\frac{\partial I(z,t)}{\partial t}$$

Taking the limit as $\Delta z \to 0$ in the above equation gives

$$-\frac{\partial V(z,t)}{\partial z} = rI(z,t) + l\frac{\partial I(z,t)}{\partial t} \tag{2.8}$$

Which is the first telegrapher's equation. By using Kirchhoff's current law (KCL) on the upper node, one gets

$$I(z,t) - I(z+\Delta z,t) - g\Delta z V(z+\Delta z,t) - c\Delta z \frac{\partial V(z+\Delta z,t)}{\partial t} = 0$$

Rearranging and taking the limit as $\Delta z \to 0$ once more yields the second telegrapher's equation

$$-\frac{\partial I(z,t)}{\partial z} = gV(z,t) + c\frac{\partial V(z,t)}{\partial t} \tag{2.9}$$

Equations (2.8) and (2.9) form a set of two coupled, first-order partial differential equations (PDEs). Together, they describe the voltage and current on a two-conductor transmission line or cable.

Multiconductor transmission line

Transmission lines will often consist of more than two conductors. Typical examples are signal cables used in electronics, three-phase power systems or power umbilicals - the topic of this thesis. In the general case, a *multiconductor transmission line* (MTL) with n conductors and a *reference conductor* is said to consist of $n+1$ conductors. The reference conductor is the conductor to which all potential differences are referenced, which is usually taken to be the neutral conductor or ground.

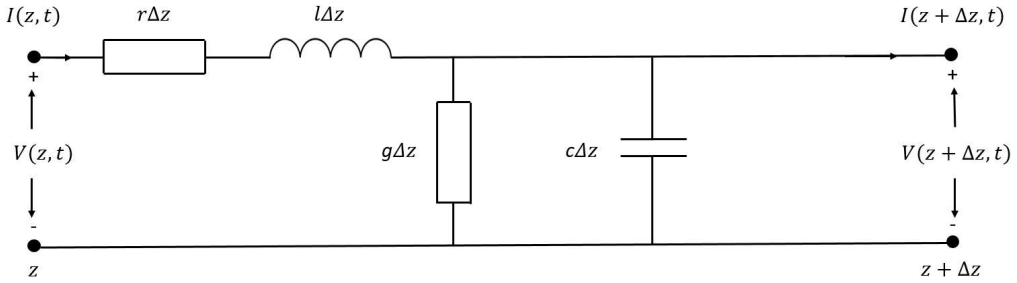


Figure 2.6: Distributed parameters circuit model for a two-conductor transmission line.

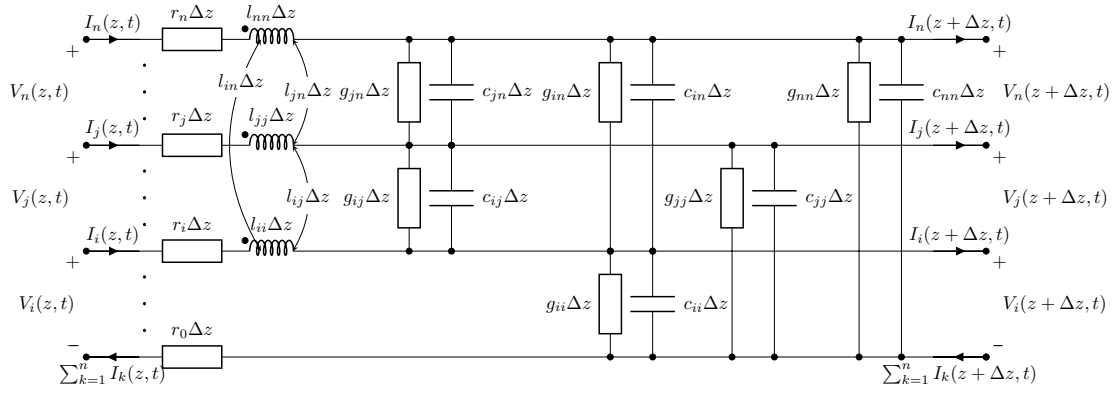


Figure 2.7: Distributed parameters circuit model for a MTL with $1, 2, ..., i, ..., j, ..., n$ conductors and a reference conductor.

Not all of these conductors are excited by an external voltage source, but due to electromagnetic induction, conduction current leakage and capacitive effects, currents and voltages can arise in elements that were not intended to carry such. This phenomenon is known as *crosstalk*.

A distributed circuit model for a multiconductor transmission line with $n+1$ conductors is shown in Fig. 2.7 above. The i-th conductor has a series impedance consisting of a resistance $r_i\Delta z$, inductances $l_{ij}\Delta z$ and a shunt admittance consisting of conductances $g_{ij}\Delta z$ and capacitances $c_{ij}\Delta z$, with $j = 1, 2, ..., n$. The reference conductor has a resistance $r_0\Delta z$. Using KVL on the outer loop consisting of the i-th conductor and the reference conductor, gives

$$V_i(z,t) - r_i\Delta z I_i(z,t) - \sum_{k=1}^{n} l_{ik}\Delta z\frac{\partial I_k}{\partial t} - V_i(z+\Delta z,t) - r_0\Delta z\sum_{k=1}^{n} I_k = 0$$

Rearranging, dividing by Δz and taking the limit as $\Delta z \to 0$ gives

$$\frac{\partial V_i(z,t)}{\partial z} = -r_i I_i(z,t) - \sum_{k=1}^{n}(r_0 I_k(z,t) + l_{ik}\frac{\partial I_k(z,t)}{\partial t})$$

which is the first telegrapher's equation for the i-th conductor in Fig. 2.7. Then applying KCL to the i-th conductor

$$\begin{aligned} I_i(z,t) - I_i(z+\Delta z,t) - \sum_{\substack{k=1\\k\neq i}}^{n} g_{ik}\Delta z(V_i(z,t) - V_k(z,t)) - g_{ii}\Delta z V_i(z,t) \\ - \sum_{\substack{k=1\\k\neq i}}^{n} c_{ik}\Delta z\frac{\partial(V_i(z,t) - V_k(z,t))}{\partial t} - c_{ii}\Delta z\frac{\partial V_i(z,t)}{\partial t} = \\ I_i(z,t) - I_i(z+\Delta z,t) - \sum_{k=1}^{n} g_{ik}\Delta z V_i(z,t) + \sum_{\substack{k=1\\k\neq i}}^{n} g_{ik}\Delta z V_k(z,t) \\ - \sum_{k=1}^{n} c_{ik}\Delta z\frac{\partial V_i(z,t)}{\partial t} + \sum_{\substack{k=1\\k\neq i}}^{n} c_{ik}\Delta z\frac{\partial V_k(z,t)}{\partial t} = 0 \end{aligned}$$

Following the same procedure as for the first telegrapher's equation for the i-th conductor, one gets

$$\frac{\partial I(z,t)}{\partial t} = -\sum_{k=1}^{n} g_{ik}V_i(z,t) + \sum_{\substack{k=1\\k\neq i}}^{n} g_{ik}V_k(z,t) - \sum_{k=1}^{n} c_{ik}\frac{\partial V_i(z,t)}{\partial t} + \sum_{\substack{k=1\\k\neq i}}^{n} c_{ik}\frac{\partial V_k(z,t)}{\partial t}$$

which is the second telegrapher's equation for the i-th conductor in Fig. 2.7. The MTL telegrapher's equations describing the voltages and currents on the n conductors can be written in matrix notation as

$$-\frac{\partial \mathbf{V}(z,t)}{\partial t} = (\mathbf{R} + \mathbf{L}\frac{\partial}{\partial t})\mathbf{I}(z,t) \tag{2.10}$$

$$-\frac{\partial \mathbf{I}(z,t)}{\partial t} = (\mathbf{G} + \mathbf{C}\frac{\partial}{\partial t})\mathbf{V}(z,t) \tag{2.11}$$

where $\mathbf{V} = (V_1\ V_2\ ...\ V_n)^t$, $\mathbf{I} = (I_1\ I_2\ ...\ I_n)^t$ and the matrices are given as

$$\mathbf{R} = \begin{bmatrix} (r_1 + r_0) & r_0 & \cdots & r_0 \\ r_0 & (r_2 + r_0) & \cdots & r_0 \\ \vdots & \vdots & \ddots & \vdots \\ r_0 & r_0 & \cdots & (r_n + r_0) \end{bmatrix} \tag{2.12}$$

$$\mathbf{L} = \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1n} \\ l_{12} & l_{22} & \cdots & l_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{1n} & l_{2n} & \cdots & l_{nn} \end{bmatrix} \tag{2.13}$$

$$\mathbf{G} = \begin{bmatrix} \sum_{k=1}^{n} g_{1k} & -g_{12} & \cdots & -g_{1n} \\ -g_{12} & \sum_{k=1}^{n} g_{2k} & \cdots & -g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -g_{1n} & -g_{2n} & \cdots & \sum_{k=1}^{n} g_{nk} \end{bmatrix} \tag{2.14}$$

$$\mathbf{C} = \begin{bmatrix} \sum_{k=1}^{n} c_{1k} & -c_{12} & \cdots & -c_{1n} \\ -c_{12} & \sum_{k=1}^{n} c_{2k} & \cdots & -c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -c_{1n} & -c_{2n} & \cdots & \sum_{k=1}^{n} c_{nk} \end{bmatrix} \tag{2.15}$$

since $\mathbf{L}$, $\mathbf{G}$ and $\mathbf{C}$ are symmetric for *isotropic* media, as can be proven by energy conservation (Paul 2008). Equations (2.10) and (2.11) will for $n + 1$ conductor be a set of $2n$ coupled first-order partial differential equations, with $\mathbf{R}$, $\mathbf{L}$, $\mathbf{G}$ and $\mathbf{C}$ being of dimensions $n \times n$.

Applicability of the telegrapher's equations

The distributed parameters circuits shown in Fig. 2.6 and 2.7 were presented without giving possible underlying assumptions any thought. As always, it is important to know the limits of the governing mathematical model, and so the applicability of the telegrapher's equations will be discussed briefly.

The fundamental underlying assumption of the telegrapher's equations is that the electromagnetic fields surrounding the conductors in a transmission line have a *transverse electromagnetic* (TEM) structure, i.e. that the electric- and magnetic fields parallel to the conductors are zero. For certain ideal transmission lines consisting of conductors with infinite conductivity, this can be shown to be exact. Although it is an assumption for real conductors, many transmission line structures can be said to propagate in TEM mode without introducing a significant error.

One could analyze transmission lines without assuming a TEM field structure, but this would mean one would have to abandon the circuit-analysis concept completely, because one would not be able to uniquely define voltages and currents in transmission lines. For example, for the voltage between two conductors to be uniquely determined, the line integral

$$\int_C \mathbf{E} \cdot d\mathbf{l} \tag{2.16}$$

would have to be path-independent for any curve C in a certain transverse plane, i.e. the electric field $\mathbf{E}$ would have to be equal to the gradient of the electric potential V. This is not the case if, for example, a time-varying longitudinal magnetic field exists, because this would add to the electric field a term according to Faraday's law.

If a line is expected to support a TEM field structure, the assumptions $\mathbf{E}_z = 0$ and $\mathbf{B}_z = 0$ outside the conductors are imposed, and it can be shown that the transverse plane fields $\mathbf{E}_{xy}$ and $\mathbf{B}_{xy}$ fullfill Maxwell's equations for static fields, and so voltages and currents *can* be uniquely defined (Ramo, Whinnery, and Duzer 1994).

To assure that the the mode of propagation in a transmission line is actually TEM or close to it, the cross section of the transmission line must be *electrically small* for reasons related to retardation. In short, this means that the largest dimension of a transmission line cross-section, for example conductor spacing or radii of armour, must be much smaller than the wavelength of the propagating TEM fields. For practical transmission line designs, this is usually achieved with negligible errors (Paul 2008).

There are, in addition to the above, a number of other properties of real-world transmission lines that violate the basic assumption of a TEM field structure, but it is generally assumed that these violations do not introduce significant errors.

2.2.3 General solution in frequency domain

Consider the currents and voltages exciting a transmission line to be time-harmonic, that is, they are on the form

$$\mathbf{V}(z,t) = \mathrm{Re}\{\mathbf{v}(z)e^{j\omega t}\}$$
$$\mathbf{I}(z,t) = \mathrm{Re}\{\mathbf{i}(z)e^{j\omega t}\}$$

where the entries in $\mathbf{v}(z)$ and $\mathbf{i}(z)$ are phasor quantities, j is the imaginary constant, ω is the electrical angular frequency and t is time. In this case it can be beneficial to Fourier-transform Eqs. (2.10) and (2.11) and analyze the set of equations in frequency domain. The telegrapher's equations in the frequency domain are given as

$$-\frac{d\mathbf{v}(z)}{dz} = \mathbf{Z}\mathbf{i}(z) \tag{2.17}$$
$$-\frac{d\mathbf{i}(z)}{dz} = \mathbf{Y}\mathbf{v}(z) \tag{2.18}$$

where $\mathbf{Z}$ and $\mathbf{Y}$ are the impedance- and admittance matrix, respectively, given as

$$\mathbf{Z} = \mathbf{R} + j\omega\mathbf{L} \tag{2.19}$$
$$\mathbf{Y} = \mathbf{G} + j\omega\mathbf{C} \tag{2.20}$$

Equations (2.17) and (2.18) form a set of $2n$ coupled first-order ordinary differential equations (ODEs). The two equations can easily be decoupled from each other. By differentiating (2.17) and substituting into (2.18) and vise versa for (2.18) into (2.17) leads to the result

$$-\frac{d^2\mathbf{v}(z)}{dz^2} = \mathbf{ZY}\mathbf{v}(z) \tag{2.21}$$
$$-\frac{d^2\mathbf{i}(z)}{dz^2} = \mathbf{YZ}\mathbf{i}(z) \tag{2.22}$$

where it is assumed that $\mathbf{Z}$ and $\mathbf{Y}$ are invariant of z, meaning that Eqs. (2.21) and (2.22) are valid only for uniform lines. Note that although (2.21) is decoupled from (2.22), the current and voltage of a conductor are not yet decoupled from the currents and voltages of other conductors. Motivated by the wish to completely decouple the currents and voltages on the conductors, modal matrices $\mathbf{T}_V$ and $\mathbf{T}_I$ of dimensions $n \times n$ are introduced in the transformations

$$\mathbf{v}(z) = \mathbf{T}_V\mathbf{v}_m(z)$$
$$\mathbf{i}(z) = \mathbf{T}_I\mathbf{i}_m(z)$$

where subscript m denote *mode* voltages and currents. By substituting these transformations into Eqs. (2.21) and (2.22), Paul (2008) shows that the general solutions to Eqs. (2.21) and (2.22) are given by

$$\mathbf{v}(z) = \mathbf{T}_V(\mathbf{e}^{-\Lambda z}\mathbf{v}_m^+ + \mathbf{e}^{\Lambda z}\mathbf{v}_m^-) \tag{2.23}$$
$$\mathbf{i}(z) = \mathbf{T}_I(\mathbf{e}^{-\Lambda z}\mathbf{i}_m^+ - \mathbf{e}^{\Lambda z}\mathbf{i}_m^-) \tag{2.24}$$

where $\mathbf{e}^{\pm\Lambda z}$ is the *matrix exponential* and Λ^2 a diagonal matrix

$$\Lambda^2 = \mathbf{T}_I^{-1}\mathbf{YZT}_I = \mathbf{T}_V^{-1}\mathbf{ZYT}_V \tag{2.25}$$

i.e. the columns of $\mathbf{T}_V$ and $\mathbf{T}_I$ contain the eigenvectors of $\mathbf{ZY}$ and $\mathbf{YZ}$ respectively, and Λ^2 contain their eigenvalues.

The general solution presented above needs $4n$ boundary conditions to form a specific solution, which are represented by $\mathbf{v}_m^\pm$ and $\mathbf{i}_m^\pm$. A relation between the voltages and currents at the boundaries can be established, and so the number of boundary conditions can be reduced to $2n$. By substituting Eq. (2.24) into Eq. (2.18) and left multiplying with $\mathbf{Y}^{-1}$ yields

$$\mathbf{v}(z) = \mathbf{Z}_c\mathbf{T}_I(\mathbf{e}^{-\Lambda z}\mathbf{i}_m^+ + \mathbf{e}^{\Lambda z}\mathbf{i}_m^-)$$

where $\mathbf{Z}_c$ is the defined to be the *characteristic impedance matrix* of MTLs and is equal to

$$\mathbf{Z}_c = \mathbf{Y}^{-1}\mathbf{T}_I\Lambda\mathbf{T}_I^{-1} \tag{2.26}$$

so the final form of the solution to the MTL telegrapher's equations and the one that will be used onward is

$$\mathbf{v}(z) = \mathbf{Z}_c\mathbf{T}_I(\mathbf{e}^{-\Lambda z}\mathbf{i}_m^+ + \mathbf{e}^{\Lambda z}\mathbf{i}_m^-) \tag{2.27}$$

$$\mathbf{i}(z) = \mathbf{T}_I(\mathbf{e}^{-\Lambda z}\mathbf{i}_m^+ - \mathbf{e}^{\Lambda z}\mathbf{i}_m^-) \tag{2.28}$$

with $2n$ unspecified boundary conditions. How to specify and implement those will be dealt with in the upcoming section.

2.2.4 Boundary conditions

To form a specific solution to the MTL equations, one needs to specify and include the boundary or terminal conditions. That is, specify what happens in terms of currents and voltages at both ends of the MTL. This is done by representing both the sources and loads as Thévenin equivalents, by using Thévenin's theorem.

Given a general MTL with $n+1$ conductors and length $\boldsymbol{\ell}$, we denote the sending end of the MTL as the *source* end, while the receiving end is denoted the *load* end. This notion is just for practical purposes - both ends can contain electrical machines and both ends can contain electrical loads. Let the source be placed at $z = 0$, while the load is placed at $z = \boldsymbol{\ell}$. Then the $2n$ source- and load Thévenin equivalents are given as

$$\mathbf{v}(z = 0) = \mathbf{v}_S - \mathbf{Z}_S\mathbf{i}(z = 0) \tag{2.29}$$

$$\mathbf{v}(z = \boldsymbol{\ell}) = \mathbf{v}_L + \mathbf{Z}_L\mathbf{i}(z = \boldsymbol{\ell}) \tag{2.30}$$

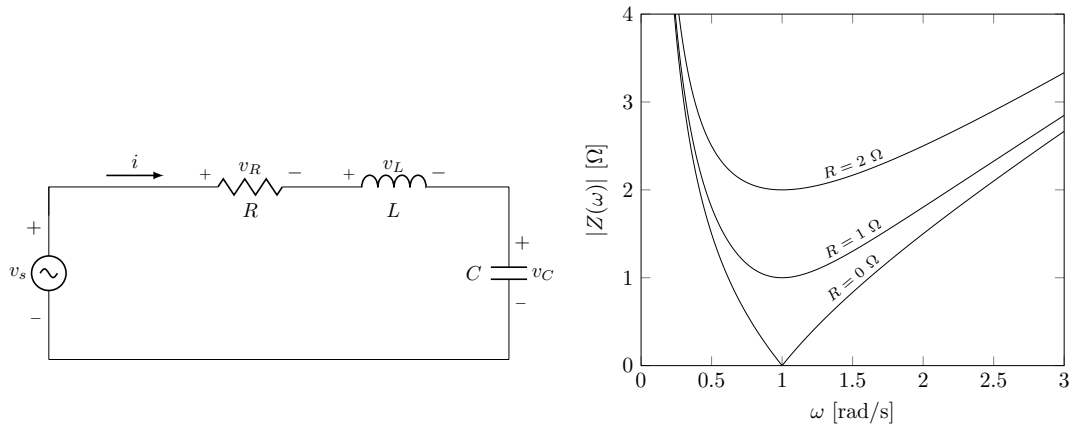
where the $n \times 1$ vector $\mathbf{v}_S$ contain the source voltages and the $n \times n$ matrix $\mathbf{Z}_S$ is the impedance matrix for the source. The second equation with subscript L is for the load, and is in full analogy to the source Thévenin equivalent. By evaluating Eqs. (2.27) and (2.28) at $z = 0$ and $z = \boldsymbol{\ell}$ and substitute the results into Eqs. (2.29) and (2.30) one gets the matrix equations contained in one system as

$$\begin{bmatrix} (\mathbf{Z}_C + \mathbf{Z}_S)\mathbf{T}_I & (\mathbf{Z}_C - \mathbf{Z}_S)\mathbf{T}_I \\ (\mathbf{Z}_C - \mathbf{Z}_L)\mathbf{T}_I\mathbf{e}^{-\Lambda\boldsymbol{\ell}} & (\mathbf{Z}_C + \mathbf{Z}_L)\mathbf{T}_I\mathbf{e}^{\Lambda\boldsymbol{\ell}} \end{bmatrix} \begin{bmatrix} \mathbf{i}_m^+ \\ \mathbf{i}_m^- \end{bmatrix} = \begin{bmatrix} \mathbf{v}_S \\ \mathbf{v}_L \end{bmatrix} \tag{2.31}$$

as shown in (Paul 2008). Solving these matrix equations yields the necessary $2n$ boundary conditions. If, on the other hand, one wishes to model the load as current sources, Northon equivalents can be used. By following the same procedure as above, the $2n$ boundary conditions can be found by simultaneously solving

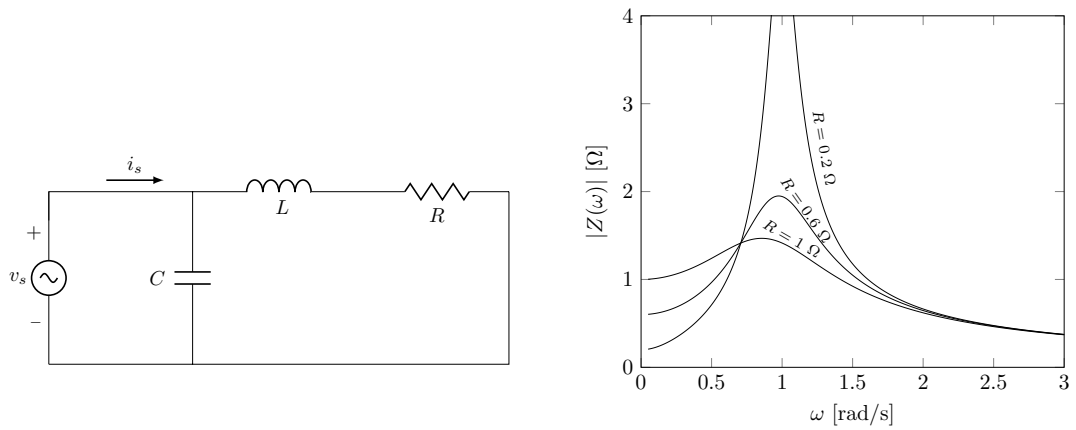
$$\begin{bmatrix} (\mathbf{Y}_S\mathbf{Z}_C + \mathbf{I}_n)\mathbf{T}_I & (\mathbf{Y}_S\mathbf{Z}_C - \mathbf{I}_n)\mathbf{T}_I \\ (\mathbf{Y}_L\mathbf{Z}_C - \mathbf{I}_n)\mathbf{T}_I\mathbf{e}^{-\Lambda\boldsymbol{\ell}} & (\mathbf{Y}_L\mathbf{Z}_C + \mathbf{I}_n)\mathbf{T}_I\mathbf{e}^{\Lambda\boldsymbol{\ell}} \end{bmatrix} \begin{bmatrix} \mathbf{i}_m^+ \\ \mathbf{i}_m^- \end{bmatrix} = \begin{bmatrix} \mathbf{i}_S \\ \mathbf{i}_L \end{bmatrix} \tag{2.32}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix, $\mathbf{Y}_S$ and $\mathbf{Y}_L$ are the source- and load admittance matrices, and $\mathbf{i}_S$ and $\mathbf{i}_L$ contain the source- and load current sources, respectively.



(a) Series RLC circuit. **(b)** Series RLC circuit impedance.

Figure 2.8: Series RLC circuit and impedance plot with $L = 1$ H and $C = 1$ F. R as shown in (b).



(a) Parallel RLC circuit **(b)** Parallel RLC circuit impedance.

Figure 2.9: Parallel RLC circuit and impedance plot with $L = 1$ H and $C = 1$ F. R as shown in (b).

2.3 Other relevant topics

2.3.1 RLC circuits and resonance

A phenomenon existing in circuits with a resistance R, inductance L and capacitance C is *resonance*. Since a power cable consists of all three elements (in a distributed form), resonance will occur at certain frequencies. In electrical circuit theory, one differentiates between *series-* and *parallel resonance*. Analyzing the simple RLC circuits shown in Fig. 2.8 and 2.9 can aid in building intuition and understanding of more complex resonance phenomena in cables.

By plotting the *input impedance*, i.e. the impedance seen by the source, as a function of frequency, one can clearly see the effect of resonance. Series resonance gives a decrease in network impedance, while parallel resonance gives an increase.

For a constant current driven parallel circuit, Ohm's law predicts that voltage amplification will occur at resonance, and visa versa for current amplification in a constant voltage driven series circuit.

The circuits of Fig. 2.8 and 2.9 have only one resonance frequency. On the other hand, a power cable can bee seen as a interconnection of an infinite amount of RLC elements in a distributed parameters model, and will therefore have an infinite amount of resonance frequencies.

Even though the cables are only excited by a fundamental frequency, higher order current harmonics can occur. This means that if current harmonics of a certain order corresponds to e.g. a parallel resonance frequency of the cable, voltage amplification may happen.

By noting that resonance occur when the phase angle is zero, it can be shown that the resonance frequency for the series circuit is

$$f_{res} = \frac{1}{2\pi}\frac{1}{\sqrt{LC}} \tag{2.33}$$

while for the parallel circuit, resonance occurs when

$$f_{res} = \frac{1}{2\pi}\sqrt{\frac{1}{LC} - \left(\frac{R}{L}\right)^2} \tag{2.34}$$

and thus, for a constant resistance, an increase in L and C means that resonance will occur at lower frequencies.

2.3.2 Skin effect

As mentioned briefly in section 2.2.1, a conductor carrying an ac current will have a non-uniform current distribution, with increasingly more of the current flowing in the out most part of the conductor. A mathematical treatment of this phenomenon will be given in this section.

In Appendix A, it is shown that the longitudinal electric field inside a circular conductor with radius R is described by the modified Bessel's equation

$$\frac{1}{\rho}\frac{d}{d\rho}\Big[\rho\frac{d(E_z(\rho))}{d\rho}\Big] - m^2 E_z(\rho) = 0 \tag{2.35}$$

where

$$m = \sqrt{j\omega\mu\sigma} = \frac{1+j}{\delta} \tag{2.36}$$

is the reciprocal of the *complex penetration depth* of the conductor, and δ is the *skin depth* equal to

$$\delta = \sqrt{\frac{2}{\omega\mu\sigma}} \tag{2.37}$$

To give some physical intuition about the skin depth, the outer part of the conductor can be studied, where $\rho \approx R$. The modified Bessel's equation then reduces to the ODE

$$\frac{d^2(E_z(\rho))}{d\rho^2} - m^2 E_z(\rho) = 0 \tag{2.38}$$

which has the general solution

$$E_z(\rho) = Ae^{-m\rho} + Be^{m\rho} \tag{2.39}$$

or, approximately, since $m >> 1$ in a good conductor for high frequencies

$$E_z(\rho) \approx Be^{m\rho} \tag{2.40}$$

where B is determined from the surface value of the electric field, so that the electric field inside the conductor near the surface is given by

$$E_z(\rho) = E|_s e^{m(\rho-R)} = E|_s e^{\frac{1+j}{\delta}(\rho-R)} \tag{2.41}$$

or, for the current density, since $J_z = \sigma E_z$

$$J_z(\rho) = J|_s e^{\frac{1+j}{\delta}(\rho-R)} \tag{2.42}$$

and therefore, at a depth equal to one skin depth, $\rho = R - \delta$, gives the value for the current density

$$J_z(R-\delta) = J|_s \cdot 1/e^{1+j} \tag{2.43}$$

and so the skin depth of the conductor is the depth at which the current density (and also the electric field intensity) has fallen to a value of $1/e$ of its surface value.

The skin depth is dependent on the reciprocal of the square root of the frequency, as given by Eq. (2.37), and hence the current will be increasingly more confined to the outer part of the conductor as frequency increases.

2.3.3 Surface impedance

As discussed in the previous section, current migrates towards the surface of the conductor when frequency increases. Since the current density is equal to $\sigma\mathbf{E}$, this leads to the conclusion that the electric field vanishes in the interior of a conductor when frequency increases.

For high frequencies, current will reside purely on the surface of the conductor. To have a definition of impedance that is applicable for all frequencies, the term *surface impedance* is introduced. The per unit length surface impedance is defined as the ratio of the electric field intensity at the surface $\mathbf{E}|_s$ of the conductor to the total current I carried by the conductor,

$$\mathrm{z} = \frac{\mathbf{E}|_s}{I} \tag{2.44}$$

3. Electromagnetic modelling of power umbilical systems

The overall goal of this chapter is to utilize the analytical solution of the multiconductor transmission line telegrapher's equations, given by Eqs. (2.27) and (2.28) as

$$\mathbf{v}(z) = \mathbf{Z}_c\mathbf{T}_I(\mathbf{e}^{-\Lambda z}\mathbf{i}_m^+ + \mathbf{e}^{\Lambda z}\mathbf{i}_m^-)$$
$$\mathbf{i}(z) = \mathbf{T}_I(\mathbf{e}^{-\Lambda z}\mathbf{i}_m^+ - \mathbf{e}^{\Lambda z}\mathbf{i}_m^-)$$

to calculate voltages and currents at each z coordinate along the cable length for various frequencies f. The mathematical aspect of finding a specific solution when $\mathbf{Z}$, $\mathbf{Y}$ and boundary conditions are known is a relatively straight forward implementation of the theory presented in section 2.2.3. Therefore, it will only be dealt with when presenting algorithms in Chapter 4.

The main challenge in cable modelling is determining the series impedance and shunt admittance matrices $\mathbf{Z}$ and $\mathbf{Y}$. The entries in these matrices can, for example, be found by using analytic formulas. Another approach would be using numerical methods, such as Finite-Element Methods (FEM) (Gustavsen et al. 2009) or the more recently proposed Method of Moments - Surface Operator (MoM-SO) (Patel, Gustavsen, and Triverio 2013a, Patel, Gustavsen, and Triverio 2013b, Patel and Triverio 2016). Or, naturally, the entries can also be obtained experimentally.

As briefly explained in Chapter 1, and as will be be dealt with more thoroughly in the upcoming Chapter 4, a goal for this thesis is to establish a computer program where users can construct a power umbilical by picking elements from a pre-existing library (or expand the library if new elements needs to be included). Due to the limited extent of this thesis, and since analytic formulas are available for elements in a power umbilical, an analytic approach will be taken.

The usual approach for assembling the series impedance and shunt admittance matrices from analytical formulas is credited to the classical paper from Ametani (1980), which is the formulation that will be used in this thesis. This general formulation for $\mathbf{Z}$ and $\mathbf{Y}$, as well as formulas used for their respective entries will be presented in this chapter.

Due to varying applications, some information about the nomenclature used onward should be given. Ametani (1980) studies a pipe-type cable, i.e. a tubular conducting pipe enclosing single-core cables. Since power umbilicals have armours comprised of stranded steel wires, and not tubular pipes, the term "pipe" will not be

applied. Instead, the term *surroundings* will be used, and can refer to seawater, air or any other surrounding medium. Furthermore, since a power umbilical can have constituents that are *not* single-core cables, the term *element* will be used when presenting the model, as has been done earlier in this text when describing power umbilicals.

Lastly, before the model is presented, some general simplifications and assumptions are made that one needs to be aware of:

1. Armour is neglected
2. Layers in power umbilicals must be modelled individually
3. Power umbilicals are placed in infinite, homogeneous surroundings
4. The space inside power umbilicals around the elements is homogeneous
5. Saturation of steel is neglected
6. Proximity effect between elements is neglected
7. Dielectric losses are neglected
8. Electrical quads and fiber optic elements are neglected
9. The eccentric position of elements inside the power umbilical does not affect its internal impedance
10. Conducting seawater inside flooded power umbilicals is neglected

The reader may wonder about some of these simplifications and assumptions. The 1$^{\text{st}}$ simplification is due to the pipe-type formulation of Ametani (1980) being a nonphysical formulation for the armour of power umbilicals, which is usually comprised of steel wires. The 2$^{\text{nd}}$ simplification is due to the twisting of the inner and outer layer in different directions. Since each layer consists of a balanced three phase circuit with currents summing to zero, it will not lead to a net induced current or voltage in other layers, when averaged over one pitch length. For the 5$^{\text{th}}$ simplification, the reader is referred to Ametani (1980). For the 7$^{\text{th}}$ simplification, the conductance is assumed to be of smaller significance, but can be added to the model later. For the 10$^{\text{th}}$ simplification, the reader should note the large difference in resistivity between metals ($\sim 10^{-8}$ Ωm) and seawater ($\sim 10^{-1}$ Ωm).

Note that Ametani (1980) does actually provide two formulations for pipe-type armour; one where the pipe-thickness is assumed infinite, and one where it is assumed finite. Since power umbilicals have armour made of stranded steel wires, a finite pipe-type formulation to represent the armour would be nonphysical. The infinite pipe-thickness formulation was applied instead, which is used to effectively model any media surrounding the power umbilicals such as air or seawater.

Table 3.1: Notation for radii of elements in power umbilicals.

	Power phase w/ screen	Power phase w/o screen	Steel duplex tube
r_1	core radius	core radius	0
r_1'	inner semi-con outer radius	inner semi-con outer radius	0
r_2'	insulation radius	insulation radius	0
r_2	screen inner radius	outer semi-con outer radius	tube inner radius
r_3	screen outer radius	equal to r_2	tube outer radius
r_4	equal to r_3	equal to r_2	HDPE sheath radius

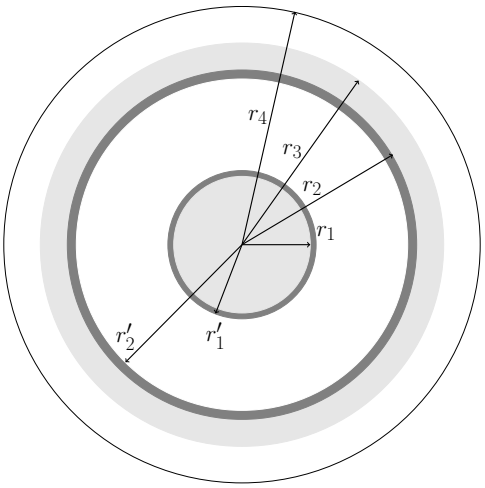


Figure 3.1: Cross section of a general element in a power umbilical with both core and metallic screen/tube. Conductors are depicted in light gray, semi-conducting layers in darker gray, and insulation/sheaths in white.

3.1 Formulation of parameter matrices

As mentioned above, the formulation of $\mathbf{Z}$ and $\mathbf{Y}$ will be that of Ametani (1980), but with a power umbilical instead of a pipe-type cable, and elements instead of single-core cables. The formulation will now be presented, by considering a set of m *general elements* placed inside a cylinder with an infinitely large outer radii, which represents the surroundings. A depiction of this with general elements i and j can be seen in Fig. 3.2, and a detailed drawing of a general element can be seen in Fig. 3.1.

The geometry of any element in a power umbilical can be derived from the one depicted in Fig. 3.1. If, for example, the screen sheath with radius r_4 is removed, the result is a power phase with a metallic screen. If the screen with outer radius r_3 is removed as well, one is left with a power phase without a metallic screen.

If one instead chooses to remove the core with radius r_1, inner semi-conducting layer with radius r_1', core insulation with radius r_2' and outer semi-conducting layer with radius r_2, one is left with a geometry representing a steel duplex tube with a HDPE sheath.

As will become clear in the following sections, the governing formulas for impedance and admittance for the different elements are numerous, but they are

also common for the different types of elements in a power umbilical. For this reason, the complete mathematical model will be presented for the general case shown in Fig. 3.1, and models for the specific elements in a power umbilical will be derived from it.

The radius shown in Fig. 3.1 will refer to different values depending on the element at interest. In Table 3.1, an overview of the notation is given.

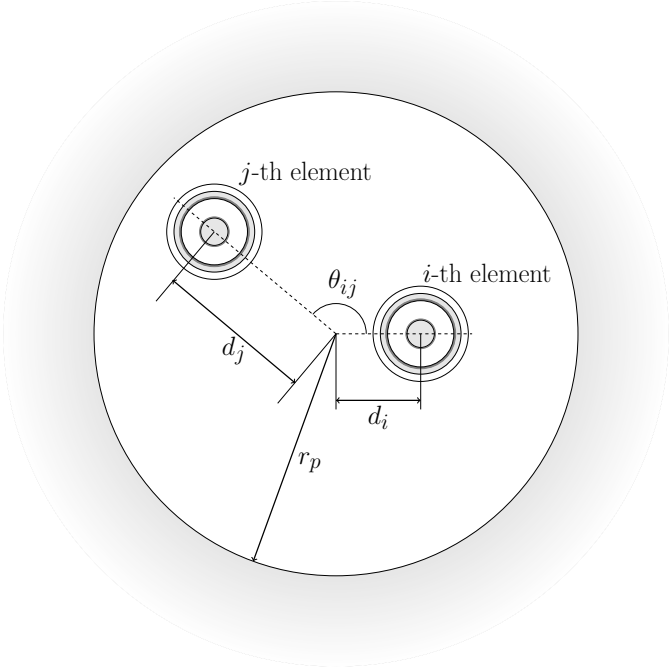


Figure 3.2: A power umbilical with outer radius r_p and with an infinite surrounding medium.

3.1.1 Series impedance matrix

For a power umbilical with m general elements as depicted in Fig. 3.2, the $n \times n$ impedance matrix can be found as the sum of two components as

$$\mathbf{Z} = \mathbf{Z}_{internal} + \mathbf{Z}_g \tag{3.1}$$

where $\mathbf{Z}_{internal}$ is the internal impedance matrix describing each element inside the power umbilical, and $\mathbf{Z}_g$ describes how the elements interact with each other with

respect to the surrounding media. They have the general form

$$\mathbf{Z}_{internal} = \begin{bmatrix} \mathbf{Z}_{int1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_{int2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{Z}_{intm} \end{bmatrix} \tag{3.2}$$

$$\mathbf{Z}_{g} = \begin{bmatrix} \mathbf{0} & \mathbf{Z}_{g12} & \dots & \mathbf{Z}_{g1m} \\ \mathbf{Z}_{g12} & \mathbf{0} & \dots & \mathbf{Z}_{g2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Z}_{g1m} & \mathbf{Z}_{g2m} & \dots & \mathbf{0} \end{bmatrix} \tag{3.3}$$

where the diagonal elements of $\mathbf{Z}_{internal}$ are sub-matrices describing each element. For the i-th element in Fig. 3.2 consisting of a core and a metallic screen/tube, the sub-matrix takes the form

$$\mathbf{Z}_{internal\ i} = \begin{bmatrix} \mathrm{z}_{cci} & \mathrm{z}_{csi} \\ \mathrm{z}_{csi} & \mathrm{z}_{ssi} \end{bmatrix} \tag{3.4}$$

where z_{cci} is the core self-impedance, z_{ssi} is the screen/tube self-impedance and z_{csi} is the mutual impedance between the core and the screen/tube. They are given as

$$\mathrm{z}_{cci} = \mathrm{z}_{c\ outer} + \mathrm{z}_{cs\ ins} + \mathrm{z}_{s\ inner} + \mathrm{z}_{s\ outer} + \mathrm{z}_{sg\ ins} + \mathrm{z}_{g\ inner} - 2\mathrm{z}_{s\ mutual} \tag{3.5}$$

$$\mathrm{z}_{ssi} = \mathrm{z}_{s\ outer} + \mathrm{z}_{sg\ ins} + \mathrm{z}_{g\ inner} \tag{3.6}$$

$$\mathrm{z}_{csi} = \mathrm{z}_{s\ outer} + \mathrm{z}_{sg\ ins} + \mathrm{z}_{g\ inner} - \mathrm{z}_{s\ mutual} \tag{3.7}$$

where all the terms are related to the i-th element and represent the following

$\mathrm{z}_{c\ outer}$ impedance of core surface

$\mathrm{z}_{cs\ ins}$ impedance of core-screen/tube insulation

$\mathrm{z}_{s\ inner}$ impedance of screen/tube inner surface

$\mathrm{z}_{s\ outer}$ impedance of screen/tube outer surface

$\mathrm{z}_{s\ mutual}$ mutual impedance between screen/tube inner and outer surface

$\mathrm{z}_{sg\ ins}$ impedance of screen/tube insulation-surrounding media

$\mathrm{z}_{g\ inner}$ impedance of surrounding media inner surface

The sub-matrices in $\mathbf{Z}_g$ describe the interaction between the elements. For the i-th and j-th element in the power umbilical, when both consists of a core and screen/tube, the sub-matrix is given as

$$\mathbf{Z}_{gij} = \begin{bmatrix} \mathrm{z}_{gij} & \mathrm{z}_{gij} \\ \mathrm{z}_{gij} & \mathrm{z}_{gij} \end{bmatrix} \tag{3.8}$$

where all entries are equal and describe the mutual impedance between the i-th and j-th element with respect to the surrounding medium. The dimensions of these

sub-matrices are dependent on the number of conductors the i-th and j-th element consists of, but the entries remains the same.

All the impedances in the above formulation are governed by analytical formulas, which will be given in the coming pages. The reader should recall that the radii denoted r_k, where $k = 1, 1', 2, 2', 3$ and 4, depend on what element is at interest, and the reader is therefore referred to Table 3.1 as guidance. The reciprocal of the complex penetration depth, introduced in section 2.3.2, is given by Eq. (2.36) as

$$m = \sqrt{j\omega\mu\sigma} = \sqrt{\frac{j\omega\mu}{\rho}}$$

The surface impedance of the core outer surface, $\mathrm{z}_{c\ outer}$, is derived in Appendix A and is given by Eq. (3.9) as

$$\mathrm{z}_{c\ outer} = \frac{\rho_c m_c}{2\pi r_1} \frac{I_0(m_c r_1)}{I_1(m_c r_1)} \tag{3.9}$$

where

ρ_c core resistivity

m_c reciprocal of core complex penetration depth

$I_n(x)$ modified Bessel function of the first kind and order n

The core-screen insulation impedance is purely due to an inductive coupling existing between currents flowing in the core and the screen. It is given by Eq. (3.10) as

$$\mathrm{z}_{cs\ ins} = \frac{j\omega\mu}{2\pi} \ln \frac{r_2}{r_1} \tag{3.10}$$

where

μ permeability of core-screen insulation

The impedance of the metallic screen/tube inner- and outer surface, and the mutual impedance between them, are derived in Schelkunoff (1934) and given by Eqs. (3.11), (3.12) and (3.13), respectively, as

$$\mathrm{z}_{s\ inner} = \frac{\rho_s m_s}{2\pi r_2} \frac{I_0(m_s r_2)K_1(m_s r_3) + I_1(m_s r_3)K_0(m_s r_2)}{I_1(m_s r_3)K_1(m_s r_2) - I_1(m_s r_2)K_1(m_s r_3)} \tag{3.11}$$

$$\mathrm{z}_{s\ outer} = \frac{\rho_s m_s}{2\pi r_3} \frac{I_0(m_s r_3)K_1(m_s r_2) + I_1(m_s r_2)K_0(m_s r_3)}{I_1(m_s r_3)K_1(m_s r_2) - I_1(m_s r_2)K_1(m_s r_3)} \tag{3.12}$$

$$\mathrm{z}_{s\ mutual} = \frac{\rho_s}{2\pi r_2 r_3} \frac{1}{I_1(m_s r_3)K_1(m_s r_2) - I_1(m_s r_2)K_1(m_s r_3)} \tag{3.13}$$

where

ρ_s screen/tube resistivity

m_s reciprocal of screen complex penetration depth

$I_n(x)$ modified Bessel function of the first kind and order n

$K_n(x)$ modified Bessel function of the second kind and order n

As was the case for the core-screen/tube insulation impedance, the screen/tube-surrounding medium insulation impedance is purely due to an inductive coupling between the screen/tube outer surface and surrounding medium. It accounts for the eccentric position of the elements and is given in F. F. d. Silva and Bak (2013) as

$$\mathrm{z}_{sg\ ins} = \frac{j\omega\mu_0}{2\pi}\Big(\mu_1 \ln\Big(\frac{r_4}{r_3}\Big) + \mu_2 \ln\Big(\frac{r_p}{r_4}[1 - \Big(\frac{d}{r_p}\Big)^2]\Big)\Big) \tag{3.14}$$

where

μ_1 relative permeability of the screen/tube sheath

μ_2 relative permeability of the insulation between the screen/tube sheath and the surrounding media

r_p power umbilical outer radius

d distance from center of power umbilical to center of element

The surface impedance of the surrounding media is derived in Brown and Rocamora (1976) and given by

$$\mathrm{z}_{ginner} = \frac{j\omega\mu}{2\pi}\left[\frac{K_0(m_p r_p)}{m_p r_p K_1(m_p r_p)} + 2\sum_{n=1}^{\infty}\Big(\frac{d}{r_p}\Big)^{2n}\frac{1}{n(1+\mu_p) + m_p r_p \frac{K_{n-1}(m_p r_p)}{K_n(m_p r_p)}}\right] \tag{3.15}$$

where

μ permeability of the surrounding medium

m_p reciprocal of surrounding medium complex penetration depth

μ_p relative permeability of the surrounding medium

r_p power umbilical outer radius

d distance from center of power umbilical to center of element

$K_n(x)$ modified Bessel function of the second kind and order n

The entries in the surrounding medium mutual impedance sub-matrices describe the mutual impedance between the i-th and j-th element in a power umbilical. They are given by Brown and Rocamora (1976) as

$$\mathrm{z}_{gij} = \frac{j\omega\mu_0}{2\pi}\Bigg[\ln\frac{r_p}{\sqrt{d_i^2+d_j^2-2d_id_j\cos\theta_{ij}}} + \frac{\mu_p}{m_pr_p}\frac{K_0(m_pr_p)}{K_1(m_pr_p)} + \sum_{n=1}^{\infty}\Big(\frac{d_id_j}{r_p^2}\Big)^n\cos(n\theta_{ij})\Big[\frac{2\mu_p}{n(1+\mu_p)+m_pr_p\frac{K_{n-1}(m_pr_p)}{K_n(m_pr_p)}}-\frac{1}{n}\Big]\Bigg] \tag{3.16}$$

where

r_p power umbilical outer radius

d distance from center of umbilical to center of element

θ angle between two conductors

μ_p relative permeability of surrounding medium

m_p reciprocal of complex penetration depth of surrounding medium

$K_n(x)$ modified Bessel function of second kind and order n

Note that in the above expression, j outside the first square bracket is the imaginary unit, whilst all other appearances of j refers to conductor j.

Sub-matrices for specific elements

While the entries for $\mathbf{Z}_g$ are given by the same formula for all types of conductors, namely Eq. (3.16) above, the sub-matrices in the internal impedance matrix $\mathbf{Z}_{internal}$ is dependent on the type of element at interest.

For a power phase with a metallic screen, the formulation of its internal impedance sub-matrix is identical to that for a general element in a power umbilical, with the exception that the first term on the right hand side of Eq. (3.14) vanishes.

For a power phase without a metallic screen, the internal impedance sub-matrix reduces to

$$\mathbf{Z}_{internal\ sub} = \big[\mathrm{z}_{cc}\big] \tag{3.17}$$

where

$$\mathrm{z}_{cc} = \mathrm{z}_{c\ outer} + \mathrm{z}_{cs\ ins} + \mathrm{z}_{sp\ ins} + \mathrm{z}_{p\ inner} \tag{3.18}$$

where, once more, the first term on the right hand side of Eq. (3.14) vanishes.

For steel duplex tubes there is no core conductor, and the internal impedance sub-matrix reduces to

$$\mathbf{Z}_{internal\ sub} = \big[\mathrm{z}_{ss}\big] \tag{3.19}$$

where

$$\mathrm{z}_{ss} = \mathrm{z}_{s\ outer} + \mathrm{z}_{sp\ ins} + \mathrm{z}_{p\ inner} \tag{3.20}$$

3.1.2 Shunt admittance matrix

The admittance matrix $\mathbf{Y}$ for a power umbilical with infinite surrounding medium can be found as

$$\mathbf{Y} = j\omega\mathbf{P}^{-1} \tag{3.21}$$

where $\mathbf{P}$ is the *potential coefficient matrix* given by

$$\mathbf{P} = \mathbf{P}_{internal} + \mathbf{P}_g \tag{3.22}$$

where $\mathbf{P}^{-1} = \mathbf{C}$, $\mathbf{P}_{internal}$ is the internal potential coefficient matrix describing each element inside the power umbilical, and $\mathbf{P}_g$ describes how the elements interact with each other through capacitive couplings.

Generally, both terms on the right hand side of Eq. (3.22) have to be considered. In power umbilicals, however, the out-most layer of each element can be considered to be grounded at any point along its length. This is due to this layer being in physical contact with a low-resistance ground path through common semi-conducting layers. Therefore, transverse electric fields exists only in the dielectrics concentric to each element, with no electric field between two elements in a power umbilical. This means there is no capacitive coupling between two elements inside a power umbilical, and the potential coefficient matrix reduces to

$$\mathbf{P} = \mathbf{P}_i \tag{3.23}$$

where $\mathbf{P}_{internal}$ is a diagonal matrix

$$\mathbf{P}_{internal} = \begin{bmatrix} \mathbf{P}_{int1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{int2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{P}_{intm} \end{bmatrix} \tag{3.24}$$

for m elements inside a pipe. For the i-th general element, the sub-matrix $\mathbf{P}_{int\ i}$ has the form

$$\mathbf{P}_{int\ i} = \begin{bmatrix} p_{ci} + p_{si} & p_{si} \\ p_{si} & p_{si} \end{bmatrix} \tag{3.25}$$

where

p_{ci} potential coefficient of the core

p_{si} potential coefficient of the screen/tube

The potential coefficients are also governed by analytic formulas. The potential coefficient of the core and the screen/tube is governed by Eqs. (3.26) and (3.27), respectively, as

$$p_{ci} = \frac{1}{2\pi\epsilon_1} \ln \frac{r_2'}{r_1'} \tag{3.26}$$

$$p_{si} = \frac{1}{2\pi\epsilon_2} \ln \frac{r_4}{r_3} \tag{3.27}$$

where

ϵ_1 permittivity of core-screen insulation

ϵ_2 permittivity of the screen/tube sheath

Note that between conductors in electrical quads, there will be capacitive couplings, and hence $\mathbf{P}_g$ is not zero. If electrical quads are to be implemented in the model at a later time, it will be necessary to formulate $\mathbf{P}_g$.

Sub-matrices for specific elements

Just as was the case for the internal impedance sub-matrices, the shape of the internal potential coefficient sub-matrices changes depending on the type of element it is describing.

For a power phase with a metallic screen produced by Nexans, there is never placed any dielectric sheath outside the screen. So for both power phases with and without screens, the internal coefficient sub-matrix reduces to only one element

$$\mathbf{P}_{internal\ sub} = \begin{bmatrix} p_c \end{bmatrix} \tag{3.28}$$

with the potential-coefficient p_c given by Eq. (3.26).

For steel duplex tubes there exists no core, and hence the potential coefficient sub-matrix in this cases reduces to

$$\mathbf{P}_{internal\ sub} = \begin{bmatrix} p_s \end{bmatrix} \tag{3.29}$$

where the potential-coefficient p_s is given by Eq. (3.27).

Since that for both power phases and for steel duplex tubes, the internal potential coefficient sub-matrices reduce to matrices of dimensions 1×1, and hence the potential-coefficient matrix $\mathbf{P}$ is diagonal. This means that instead of finding the potential-coefficient matrix and perform a matrix inversion, one can merely find the capacitance matrix directly, by noting that when $\mathbf{P}$ is a diagonal matrix

$$\mathbf{C} = \mathbf{P}^{-1} = \begin{bmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_m \end{bmatrix} \tag{3.30}$$

where the diagonal entries are the self-capacitances of the elements, given by the reciprocal of either Eq. (3.26) or (3.27) depending on element type.

3.2 Modelling of terminations

The terminations of power umbilicals are generally implemented by Eqs. (2.31) or (2.32), as presented in section 2.2.4. What equation to use will depend on the termination of a specific umbilical and on what phenomena one wishes to study. There are, perhaps, a few situations that might need specific attention.

Conducting elements in the umbilicals, such as metallic screens on power phases, duplex steel tubes, armour around fiber optic elements, etc. are either grounded or

left open-circuited at each terminations. When using the Thévenin representation in Eq. (2.31), the grounding of a conductor, for example at the sending end, is represented by $v_s = 0$ and $\mathrm{z}_s = 0$. If the conductor is left open, the impedance is taken as infinity. Since infinity is not a well-defined number, the implementation of a open-circuited element will simply involve setting the impedance equal to a number much larger than the impedance of the conductor itself.

A numerically more "refined" method would be to utilize the Northon representation implemented in Eq. (2.32) to model open-circuited elements, i.e. terminations with zero shunt admittance.

There might be situations where it is necessary to model the terminations as a mix of voltage- and current sources, but implementing this is left to future development and is beyond the scope of this thesis.

4. Computer implementation

In this chapter the computer implementation of the model from the preceding chapter will be presented. The program package, referred to as UmbSim, is written in Python 2.7 and is based partially on an object-oriented approach. Although not followed rigidly, most of the source code is written in accordance with the PEP-8 style guide. The complete source code for the program package can be found in Appendix B.

There are both up- and downsides to object-oriented programming. A loss of speed may be one of the downsides, but since the number of elements in a power umbilical are few, the number of operations needed to be done are few (for a computer, that is), and thus the computation time is expected to be relatively short.

The reason for choosing an object-oriented approach is that the mathematical model and the physicality of cable modelling fits well into several of the features of object-oriented programming.

First of all, *inheritance* is a concept that is both intuitive and will minimize code duplication in the program, due to how the elements are described in the preceding chapter. The general element in a power umbilical in Fig. 3.1 can naturally take the role as a super-class, as shown in the class structure of Fig. 4.1. That is, power phases, steel duplex tubes and all the other elements can be seen as a sub-class of the general element.

Also, due to the intuitive structure and low duplicity of object-oriented programming, it is also advantageous when it comes to maintaining or modifying existing code, or when doing further development. The UmbSim package is in no way completed by the work of this thesis, and development should be able to be picked up by other developers.

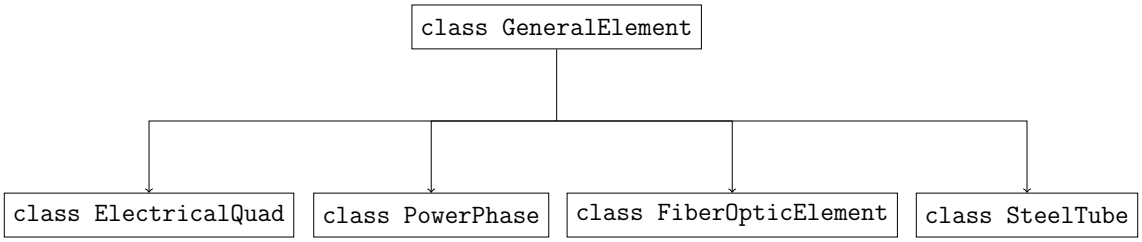


Figure 4.1: Class structure in the `system.py` module for the elements in a power umbilical.

As the proposal for the topic of this thesis was put forth by Nexans, they had several requests on routines and functionality of the program package. Among these were

- User should be able to model any power umbilical
- User should be able to model any termination situation
- Voltage, current and impedance plots of both magnitude and angle for any element in a power umbilical, at any position and frequency should be available
- Program should include routines for calculating maximum voltage on and current in any element
- Program should include routines for studying systems with harmonic content
- Program should include routines for finding both series- and parallel resonance frequencies for impedance
- Program should include routines for running simulations from pre-determined parameter matrices, which, for example, could be obtained through measurements

4.1 The UmbSim package

The UmbSim package consists of three modules: `system.py`, `simulation.py` and `solver.py`. The most important aspects of each module, and the interplay between them, will be described in this section. Firstly, a description of how a user would inteact with UmbSim is given.

Currently, there is no graphic user interface for UmbSim. To define a certain *system* with a source, a power umbilical and a load, the user start by creating an empty Python script and importing the `system.py` module from the UmbSim package. An example of how to define a power umbilical cable with three power phases without screens and three steel duplex tubes are shown in Listing 4.1. All input values have to be given in SI-units.

When studying a specific system, the user first specifies geometrical and electromagnetic properties for the elements in the power umbilical by using the `set_parameters()` method from the `system.py` module. When parameters for the elements are set, the user can define what elements to include and the terminations at each end. Then, instances of the `Umbilical`, `Source` and `Load` classes can be created.

Docstrings are available in the source code of the UmbSim package, for a full description of the available functionality. In Appendix B the source code for UmbSim is given.

```
import cmath, math
import numpy as np
from UmbSim import system

# Define power phases, steel duplex tubes and surroundings ("pipe")
system.PowerPhase.set_parameters({'d_core': 11.5e-03,
                                  'd_inner_semi_con': 13.5e-03,
                                  'd_core_insulation': 29.5e-03,
                                  'd_outer_semi_con': 37.9e-03,
                                  'core_dc_resistance': 1.93e-04,
                                  'screen': False})

system.SteelTube.set_parameters({'d_inner': 12.7e-03,
                                 'd_outer': 15.62e-03,
                                 'd_sheath': 19.02e-03,
                                 'rho_tube': 8e-07,
                                 'steel mu_r': 32})

system.Pipe.set_parameters({'d_pipe': 189e-03,
                            'rho_pipe': 0.3,
                            'mu_r_p': 1})

# Define cable length
cable_length = 31e03

# Elements with respective positions (in polar coordinates (metres, degrees))
elements = [['power phase', (22.3e-03, 0)],
            ['power phase', (22.3e-03, 120)],
            ['power phase', (22.3e-03, 240)],
            ['steel tube', (32.34e-03, 60)],
            ['steel tube', (32.34e-03, 180)],
            ['steel tube', (32.34e-03, 300)]]

# Define phasor rotation operator a, rms phase voltage V and number of conductors n
a = cmath.exp(1j * 2 * math.pi / 3)
V = 36e03 / math.sqrt(3)
n = 6

# Source
vs = np.array(1, a ** 2, a, 0, 0, 0]).transpose() * V
Rs = Ls = np.zeros((n, n))

# Load
vl = np.zeros((n, 1))
Rl = np.diag([60, 60, 60, 0, 0, 0])
Ll = np.diag([0.3, 0.3, 0.3, 0, 0, 0])

# Create instances
umbilical = system.Umbilical(elements, cable_length)
source = system.Source(vs, Rs, Ls)
load = system.Load(vl, Rl, Ll)
```

Listing 4.1: Example of input file from user.

4.1.1 system.py

The `system.py` module is the implementation of how the *system* is described, i.e. power umbilicals, sources and loads. The main task of the `system.py` module is to calculate the per unit length impedance- and admittance matrices $\mathbf{Z}$ and $\mathbf{Y}$, as well as the termination impedance matrices $\mathbf{Z}_S$ and $\mathbf{Y}_L$.

In terms of the computer implementation, the source and load is described entirely by the `Source` and `Load` classes, respectively. The `Umbilical` class manages all elements inside a power umbilical, by creating instances for each element, as well as for the mutual couplings between elements. All these instances are systematically placed into a nested-list referred to as the *instance matrix*, and from the instance matrix, the impedance- and admittance matrices are calculated. As an example, the `system.impedance_matrix()` method in the `system.py` module follow Algorithm 1 below.

Algorithm 1: The `system.impedance_matrix()` method for finding $\mathbf{Z}$

Data: instance matrix, frequency f, number of conductors n
Result: impedance matrix $\mathbf{Z}$
1 Create $\mathbf{Z}$ as an empty $n \times n$ NumPy array
2 **forall** *elements* **in** *instance matrix* **do**
3 | Calculate impedance sub-matrix at frequency f
4 | Place entries from sub-matrix into correct entries in $\mathbf{Z}$
5 **return Z**

The algorithm is very simple, but the code is a bit more complicated due to the handling of indexes and the calculation for the impedance sub-matrices for each element.

Notice that the instances need only be created once. This is also true for the entries of the capacitance matrix, which are independent of frequency, and hence the admittance matrix is then simply calculated as $j\omega\mathbf{C}$. The entries in the impedance matrix $\mathbf{Z}$, however, are frequency dependent and needs to be calculated for each frequency.

4.1.2 solver.py

The main task of the `solver.py` module is to determine the specific solutions $\mathbf{v}(z)$ and $\mathbf{i}(z)$ for a specific system at a certain frequency. If this is done for a single frequency, it is a simple task to extend the functionality to allow for a range of frequencies to be simulated.

The solution at frequency f is mainly computed through the `solver.solution()` method. The method is an implementation of Algorithm 2, which follows directly from the theory presented in Chapter 2.

The 1st step in Algorithm 2 is accomplished by using the NumPy routine for linear algebra, with the method `np.linalg.eig(A)` which takes a square array `A` as input and returns the eigenvalues and right-hand side eigenvectors. The number

Algorithm 2: The `solver.solution()` method for finding $\mathbf{v}(z)$ and $\mathbf{i}(z)$

Data: $\mathbf{Z}$, $\mathbf{Y}$, $\mathbf{Z}_S$, $\mathbf{Z}_L$, $\mathbf{v}_S$, $\mathbf{v}_L$ and $\boldsymbol{\ell}$ as input from `system.py`
Result: $\mathbf{i}$, $\mathbf{v}$ and/or $\mathbf{z}$ as NumPy arrays
1 Eigendecompose $\mathbf{YZ}$ to find $\mathbf{T}_I$ and Λ^2
2 Determine the $2n$ coefficients contained in $\mathbf{i}_m^+$ and $\mathbf{i}_m^-$ from Eq. (2.31)
3 Evaluate $\mathbf{v}(z)$ and/or $\mathbf{i}$(z) in Eqs. (2.27) and (2.28), respectively, for each desired z in $0 \leq z \leq \boldsymbol{\ell}$
4 **return** $\mathbf{i}$, $\mathbf{v}$ or $\mathbf{z}$

of columns in the returned arrays $\mathbf{i}$, $\mathbf{v}$ or $\mathbf{z}$ are determined by the number of conductors in the system, whilst the number of rows is determined by the number of z coordinates the user wishes to compute the solution for.

4.1.3 simulation.py

The main task of the `simulation.py` module is to provide users access to output data from specific solutions, either through NumPy arrays or visualizing plots.

When running simulations, a user can choose between solutions for current, voltage or impedance. This can done for either 1) A single frequency for a number of desired positions along the power umbilical, or for 2) A single position and for a number of frequencies. For example, to plot the input impedance as seen by the source, a user could add the following code to Listing 4.1.

```
51 # Continued from Listing 4.1
52
53 from UmbSim import simulation
54
55 # Simulation instance
56 sim = simulation.Simulation(umb, source_inst, load_inst)
57
58 # Define frequencies at interest and position
59 f_array = np.linspace(0, 5000, 200)
60 z = 0
61
62 # Get input impedance spectrum for power phase 1 (element no. 0)
63 impedance = sim.plot_solution(f_array, z, 0, 'z')
```

Listing 4.2: Code for simulating the input

In the above listing, the solution was requested through the method

```
sim.plot_solution(frequency array, position, element, type)
```

where `element` refers to an element number corresponding to an index in the `elements` list defined in Listing 4.1. Since Python counts from 0, valid inputs in this example would be a number from 0 to 5. The `type` argument is a string, being either `'v'`, `'i'` or `'z'`, depending on whether the solution for voltage, current or impedance is wanted.

5. Example problems and validation

Both an electromagnetic model for power umbilicals and an accompanying computer program, named UmbSim, have been presented in the two preceding chapters. A few worked example problems will now be presented, where the results from simulations in UmbSim will be validated against different methods. These methods are

1. Analytical methods
2. Flux2D
3. Measurements

Analytic methods will be applied to simple power umbilicals that exhibit a cyclic symmetric structure. This will serve a purpose of validating the implementation of the solution algorithms in the `solver.py` module. The analytical methods will use impedance and admittance values calculated by the `system.py` module, and will therefore not help validate whether the underlying model provides reasonable results.

Therefore, UmbSim results will also be compared to simulations provided by Nexans. These are simulations where the example problems are modelled and simulated in the finite element software Flux2D. As the name implies, Flux2D is only able to model two dimensional problems, but by using a combination of field- and circuit-analysis, it is possible to capture longitudinal effects. The general idea is that a long cable or power umbilical is represented as a series of several *nominal pi* equivalent circuits. To use this circuit representation, it is necessary to calculate the values of the per phase per unit length resistance and inductance in these circuits. This is achieved by finite element analysis of a cross section of the power umbilical at interest, which Flux2D does by solving the appropriate electromagnetic field equations. Then, using these obtained values in the circuit representation, voltages and currents at discrete positions along the length of the cable can be computed. For example, if a model uses a series of 10 nominal pi circuits (which is the case for the example problems in this chapter), the voltage and currents can be calculated at the sending end, the load end and at eight longitudinal positions along the cable.

As a last validation, and to test the limits of UmbSim, simulation results will be compared to measurements performed on a power umbilical. The measurements are also provided by Nexans.

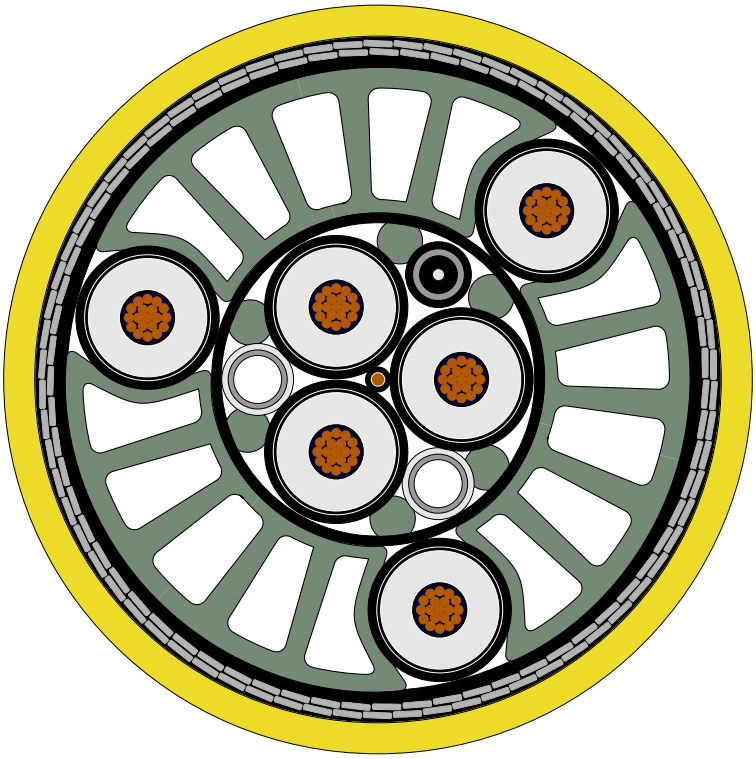


Figure 5.1: Umbilical A.

5.1 Case study - Umbilical A

The first power umbilical that will be modelled is the 31 km long Umbilical A, whose cross section depicted in Fig. 5.1. Umbilical A is twisted in two layers, naturally referred to as the inner- and outer layer. Both layers have 36 kV three-phase circuits made up by unscreened power phases with 95 mm^2 stranded copper conductors. Data for the power phases is given in Table 5.1.

The inner layer of Umbilical A additionally have two steel duplex tubes and a fiber optic element. The fiber optic element will be neglected, but data for the steel duplex tube is given in Table 5.2. The green colored parts inside Umbilical A are HDPE filler elements, which make sure that the elements stay in their respective positions.

Other general data for the umbilical, such as outer radius and data for the surrounding seawater is given in Table 5.3.

If values for μ and ϵ for specific materials are not listed in tables, they are taken as μ_0 and ϵ_0, respectively, or, they are not of interest when modelling.

The terminations for Umbilical A is given in Table 5.4. Note that the steel duplex tubes are grounded in both ends, which is common practice to avoid any high voltage from arising.

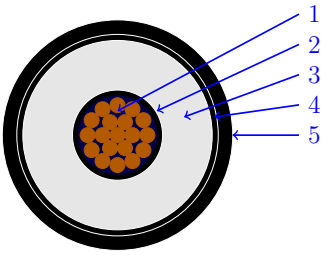


Figure 5.2: One of the power phases in Umbilical A.

Table 5.1: Data for the power phases in Umbilical A.

Power phase		Diameter [mm]
1	conductor, 95 mm^2 stranded Cu	11.5
2	semi-conducting screen, XLPE	13.5
3	insulation, XLPE	29.5
4	semi-conducting screen, XLPE	32.9
5	semi-conducting polyethylene	37.9
Electrical properties		Value
ϵ_r of XLPE		2.4
dc resistance of conductor (20 °C)		0.193 Ω/km

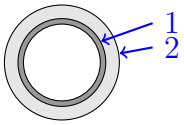


Figure 5.3: One of the steel duplex tubes in Umbilical A.

Table 5.2: Data for the steel duplex tubes in Umbilical A.

Duplex		Diameter [mm]
1	steel duplex tube, ID=12.7 mm	15.62
2	HDPE sheath	19.02
Electrical properties		Value
ϵ_r of HDPE		2.3
μ_r of duplex steel		32
Resistivity of duplex steel (20 °C)		$8 \cdot 10^{-7}$ Ωm

Table 5.3: Various data for Umbilical A and surroundings.

Umbilical A	
Length	31 km
Outer diameter	189 mm
μ inside umbilical	μ_0
Surroundings	
μ seawater	μ_0
ρ seawater	0.3 Ωm

Table 5.4: Terminations for the elements in Umbilical A.

Source end	
Power phase line-to-line voltage	36 kV
Frequency	50 Hz
Phase sequence	positive
Steel duplex tubes	grounded
R_S for power phases	0
L_S for power phases	0
Load end (nominal)	
Steel duplex tubes	grounded
R_L for power phases	60 Ω
L_L for power phases	0.3 H
No voltage sources	

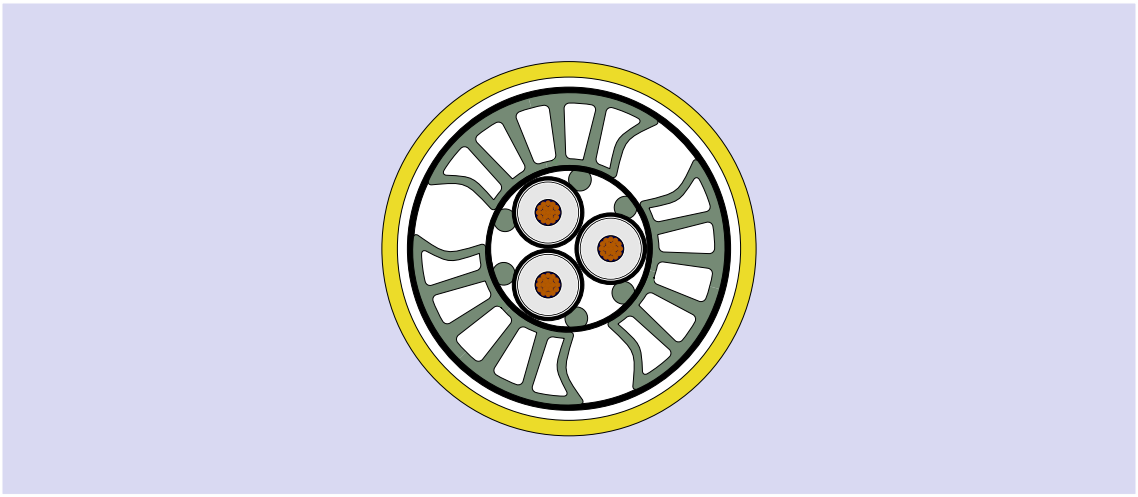


Figure 5.4: The first approach in modelling Umbilical A, referred to as Umbilical A1. Light blue represents infinite surrounding seawater.

5.1.1 Example A1

The first attempt at modelling in UmbSim will include only the inner three-phase circuit of Umbilical A. Figure 5.4 depicts a cross section of the power umbilical for this first example, referred to as Umbilical A1. Notice that due to the limits of the model, as mentioned in Chapter 3, the power umbilical will be modelled as unarmoured.

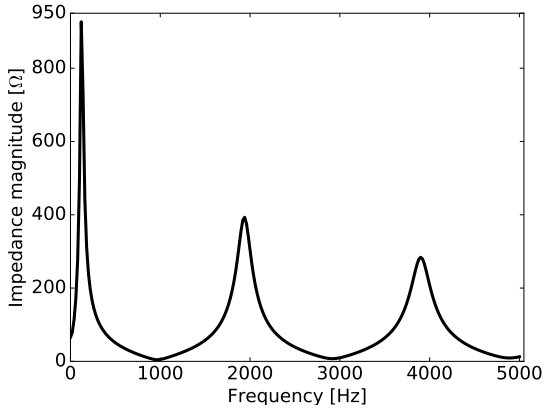
The terminations at the source- and load end of the power umbilical is given in Table 5.4. In addition to modelling with nominal load, situations where the power phases are either left open or short-circuited at the load end will be analyzed as well.

UmbSim simulation

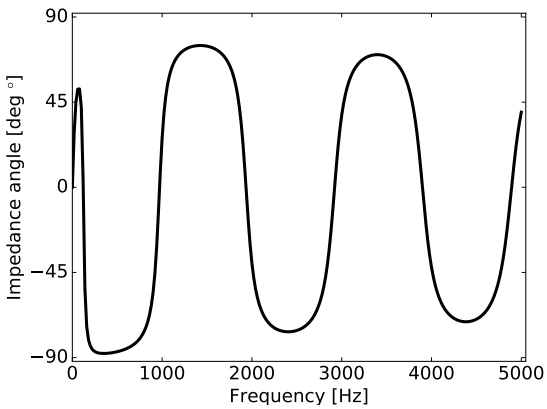
By defining Umbilical A1 and the corresponding terminations in Python scripts, simulations can be ran. The scripts are found in Appendix C.1. Figures 5.5 and 5.6 show plots for the input impedance, voltage- and current magnitudes with different load end terminations. The per phase per unit length parameters at 50 Hz are given in Table 5.6, and the dc limit impedance magnitudes for the different terminations are given in Table 5.5.

In Fig. 5.5 both series- and parallel resonance phenomena are observed. Notice that both series resonance (magnitude minima) and parallel resonance (magnitude maxima) occur at an angle equal to 0°. Also, the impedance maxima magnitudes decrease with frequency, as is in agreement with the study of the parallel RLC circuit in section 2.3.1.

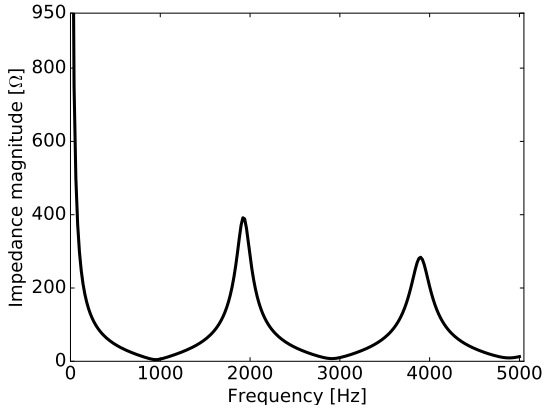
The source end voltages in Fig. 5.6 are observed to always be equal to $36/\sqrt{3}$, as they are supposed to. The voltage at the load end in the short-circuited situation is zero, and the current at the load end in the open end situation is zero, as one would expect.



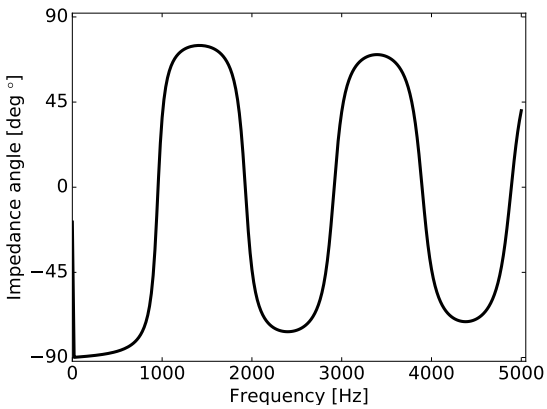
(a) Magnitude, nominal load.



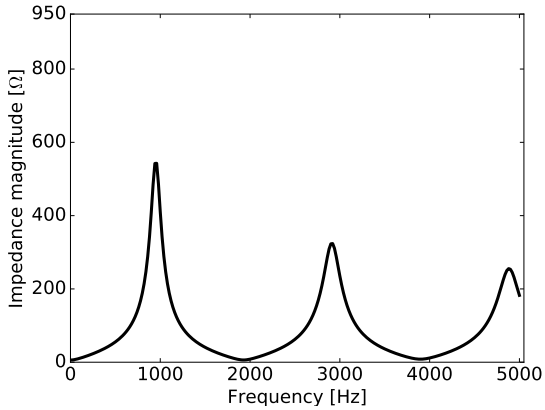
(b) Angle, nominal load.



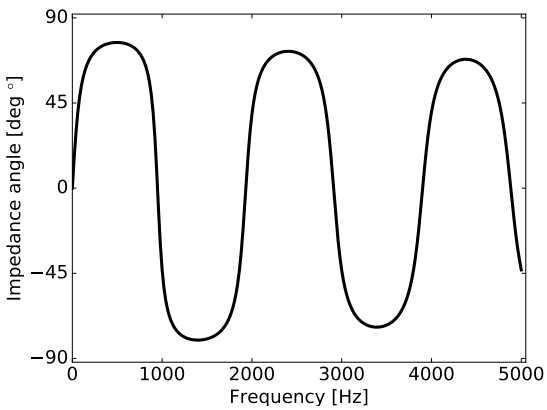
(c) Magnitude, open load end.



(d) Angle, open load end.



(e) Magnitude, shorted load end.

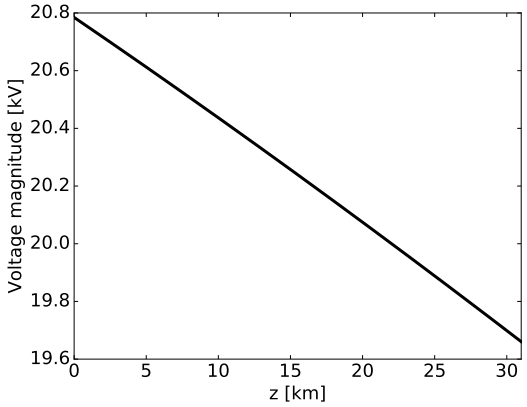


(f) Angle, shorted load end.

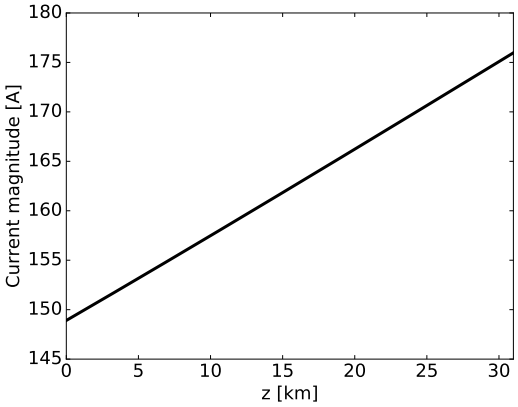
Figure 5.5: UmbSim simulations of input impedance spectra for Umbilical A1 power phases with different terminations.

Table 5.5: dc limit input impedance from Figs. 5.5a, c and e.

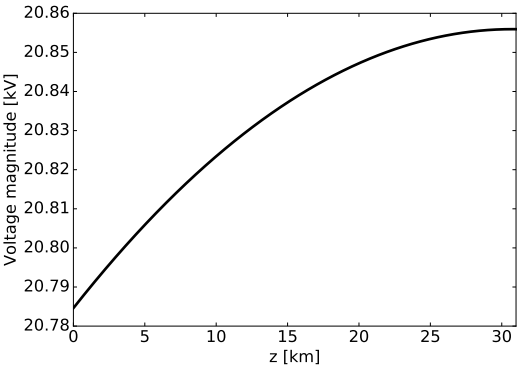
Load end termination	Input impedance [Ω]
Nominal load	65.983
Open end	10^5
Shorted end	5.983



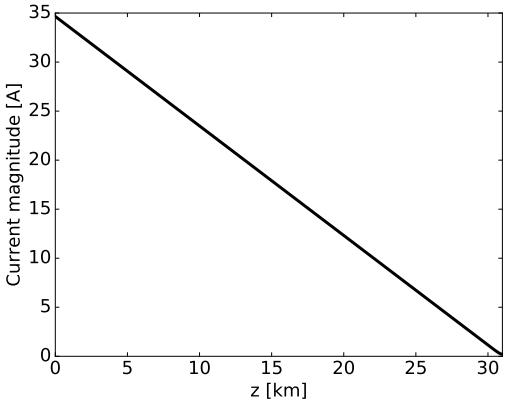
(a) Phase voltage, nominal load.



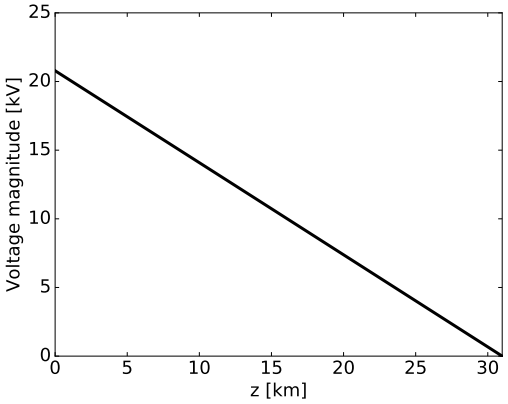
(b) Phase current, nominal load.



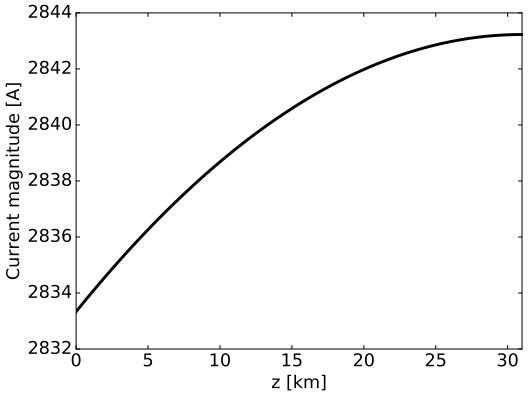
(c) Phase voltage, open load end.



(d) Phase current, open load end.



(e) Phase voltage, shorted load end.



(f) Phase current, shorted load end.

Figure 5.6: UmbSim simulations of rms voltage- and current magnitudes along power phases at 50 Hz for Umbilical A1.

Table 5.6: Per unit length parameters of Umbilical A1 at 50 Hz.

Parameter	Value
resistance	0.193 Ω/km
inductance	0.431 mH/km
capacitance	0.171 μF/km

Comparison with analytical calculations

Since this first example consist of few elements and has a symmetrical arrangement, quite a few analytical exercises can be done to help verify various aspects of UmbSim. Furthermore, since both the positive sequence voltage sources and loads are symmetric, the power umbilical can be modelled entirely by a positive sequence per phase circuit.

dc limit input impedance A first check is the value of the impedance magnitudes in the dc limit of Figs. 5.5a. c and e. The numerical values from UmbSim simulations are given in Table 5.5, which all match qualitatively well with intuition.

For the open end in Fig. 5.5c the dc limit input impedance is equal to $10^5\ \Omega$. The reason for this is that the implementation of a open-circuit termination in UmbSim is implemented as $R_L = 10^5$ and $L_L = 0$. This value is large enough, relative to impedance of the power phases, to be considered as infinite. Physically speaking, it is intuitive that the input impedance with open load end would approach infinity, as the shunt admittance goes to zero in the dc limit.

In the nominal load or shorted situation shown in Figs. 5.5a and e, the input impedance takes on finite values in the dc limit. The short-circuited situation yields the dc resistance of the conductor. By evaluating it analytically from the per unit length resistance r_{dc} from Table 5.5 multiplied with the length $\boldsymbol{\ell}$ of the power umbilical, one finds that

$$R_{dc} = r_{dc}\boldsymbol{\ell} = 5.983\ \Omega$$

which is identical to the value in Table 5.5. For the nominal load situation, one would expect the value

$$R_{dc} = r_{dc}\boldsymbol{\ell} + R_L = 65.983\ \Omega$$

which is also identical to the tabular value.

Charging current In the situation of an open load end, the only current that flows in the power phases is due to the path through the shunt admittance. The current flowing in one of the power phases is then given by the telegrapher's equation

$$-\frac{di(z)}{dz} = j\omega c v(z) \tag{5.1}$$

since there is no capacitive coupling between the power phases. Since the line is left open circuited, the current flowing at adequately low frequencies is small, and thus the voltage along the length of the cable can be assumed constant and equal to the source phase voltage v_s,

$$-\frac{di(z)}{dz} = j\omega c v_s \tag{5.2}$$

Furthermore, the boundary condition

$$i(z = \boldsymbol{\ell}) = 0 \tag{5.3}$$

must hold. Solving the ODE in Eq. (5.2) and imposing the boundary condition yields the specific solution

$$i(z) = j\omega c v_s(\boldsymbol{\ell} - z) \tag{5.4}$$

which should be valid for relatively low frequencies. The per unit length capacitance is found from Table 5.6. Then, at 50 Hz, the current drawn from the source has the value

$$i(z=0) = j\omega c v_s \boldsymbol{\ell} = j34.6 \text{ A} \tag{5.5}$$

which is purely capacitive and with a magnitude that coincides well with Fig. 5.6d. Also, from Fig. 5.12d, one sees that a low frequencies, the input impedance has a phase angle of -90 degrees, which corresponds to a capacitive reactance.

Voltage with open load end If one denotes the power phases 1, 2 and 3, the voltage $v_1(z)$ on phase 1 is given by the telegrapher's equation

$$-\frac{dv_1(z)}{dz} = \mathrm{z_s} i_1(z) + \sum_{k=2}^{3} \mathrm{z}_m i_k(z) \tag{5.6}$$

where z_s and z_m are the per unit length internal impedance of phase 1 and the mutual impedance between two phases, respectively. The mutual impedances are equal due to the power phases having equilateral spacing. Positive sequence excitation of the power phases means the currents have a 120° phase shift between them, so that

$$\begin{aligned} i_2(z) &= a^2 i_1(z) \\ i_3(z) &= a i_1(z) \end{aligned}$$

where a is the phasor rotation operator $e^{j\frac{2\pi}{3}}$. By using the relation $1 + a + a^2 = 0$, Eq. (5.6) can be written as

$$-\frac{dv_1(z)}{dz} = (\mathrm{z}_s - \mathrm{z}_m) i_1(z) = (r + j\omega l) i_1(z) \tag{5.7}$$

where r and l are the per unit length resistance and inductance from Table 5.6, respectively. By taking the derivative of the above equation and substituting in Eq. (5.1) one separates the telegrapher's equations and gets

$$\frac{d^2 v_1(z)}{dz^2} = \gamma^2 v_1(z) \tag{5.8}$$

where $\gamma = \sqrt{(r - j\omega l) j\omega c}$. Equation (5.8) has the general solution

$$v_1(z) = v^+ e^{-\gamma z} + v^- e^{\gamma z} \tag{5.9}$$

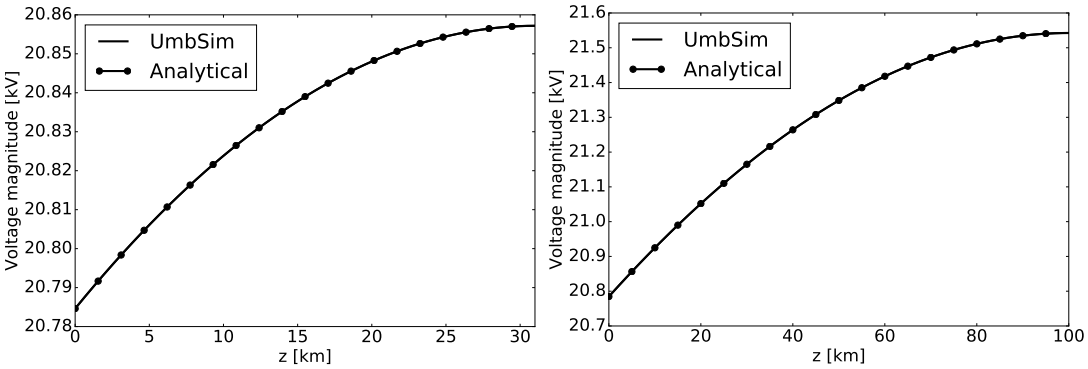
which is Eq. (2.23) in scalar form. The specific solution is found by imposing the boundary conditions

$$v_1(z=0) = v_s \tag{5.10}$$

$$\frac{dv_1(z=\boldsymbol{\ell})}{dz} = 0 \tag{5.11}$$

which holds since the load end is left open. The specific solution is then given as

$$v_1(z) = v_s \Big[\Big(1 - \frac{1}{1 + e^{2\boldsymbol{\ell}\gamma}} \Big) e^{-\gamma z} + \frac{1}{1 + e^{2\boldsymbol{\ell}\gamma}} e^{\gamma z} \Big] \tag{5.12}$$



(a) 31 km. **(b)** 100 km.

Figure 5.7: rms voltage along power phase with open load end at 50 Hz for (a) 31 km and (b) 100 km versions of Umbilical A1.

The result from the UmbSim simulation of the 31 km version of Umbilical A1 is plotted together with Eq. (5.12) in Fig. 5.7a. Since the voltage does not change much across the cable (only about a 70 V drop over 31 km), another situation should be analyzed. Equation (5.12) evaluated at $z = \boldsymbol{\ell}$ gives

$$v(z = \boldsymbol{\ell}) = 2v_s \frac{e^{\gamma \boldsymbol{\ell}}}{1 + e^{2\gamma \boldsymbol{\ell}}} \tag{5.13}$$

To find the dominating terms in the above equation a series expansion can be done. By noting that $\gamma\boldsymbol{\ell} << 1$ for low frequencies and retaining only first order terms,

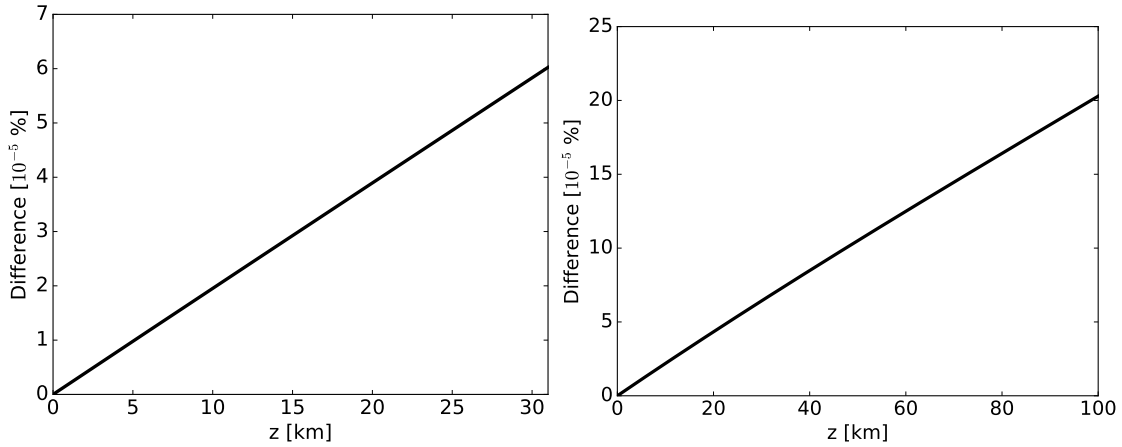
$$v(z = \boldsymbol{\ell}) \approx v_s(\frac{\gamma \boldsymbol{\ell}}{2} + 1) \tag{5.14}$$

or

$$\frac{v(z = \boldsymbol{\ell})}{v_s} \approx \frac{\gamma \boldsymbol{\ell}}{2} + 1 \tag{5.15}$$

which means that, for example, increasing the line length should yield an increase in the difference between the source- and load end voltage. UmbSim and the analytical result of Eq. (5.12) is compared in Fig. 5.7b for a power umbilical with the same cross section as Umbilical A1, but with a length of 100 km. The percentage difference of the UmbSim result with respect to the analytical result is plotted in Fig. 5.8a and b. The difference is practically zero, as it should be, since UmbSim is an implementation of the exact analytical solution.

The observed voltage rise at the receiving end of cables in open-circuited situations is a well-known phenomenon called the *Ferranti effect* (F. F. d. Silva and Bak 2013).



(a) 31 km. **(b)** 100 km.

Figure 5.8: Percentage difference between UmbSim and analytical calculations for the voltage along power phase for (a) 31 km and (b) 100 km versions of Umbilical A1. Open load end termination.

Input impedance Since it is adequate to describe the system by a per phase representation, the solution of the MTL telegrapher's equation reduces to

$$v(z) = Z_C(i^+ e^{-\gamma z} + i^- e^{\gamma z}) \tag{5.16}$$
$$i(z) = i^+ e^{-\gamma z} - i^- e^{\gamma z} \tag{5.17}$$

where Z_C is the characteristic impedance and γ the propagation constant, which are equal to

$$Z_C = \sqrt{\frac{r + j\omega l}{j\omega c}} \tag{5.18}$$
$$\gamma = \sqrt{(r + j\omega l) j\omega c} \tag{5.19}$$

under the assumption that conductance through the insulation is zero.

The input impedance as a function of frequency is found in UmbSim as $v(z = 0)/i(z = 0)$. This corresponds to the situation when

$$Z_{in}(\omega) = \frac{v(z = 0)}{i(z = 0)} = Z_C \frac{i^+ + i^-}{i^+ - i^-} \tag{5.20}$$

from evaluating Eqs. (5.16) and (5.17) at $z = 0$. The two undetermined coefficients i^+ and i^- are found from the Dirichlet boundary conditions

$$v(z = \boldsymbol{\ell}) = i_L Z_L \tag{5.21}$$
$$i(z = \boldsymbol{\ell}) = i_L \tag{5.22}$$

where i_L is the current flowing through the load impedance Z_L. Substituting these boundary conditions into the general solutions of Eqs. (5.16) and (5.17), the unde-

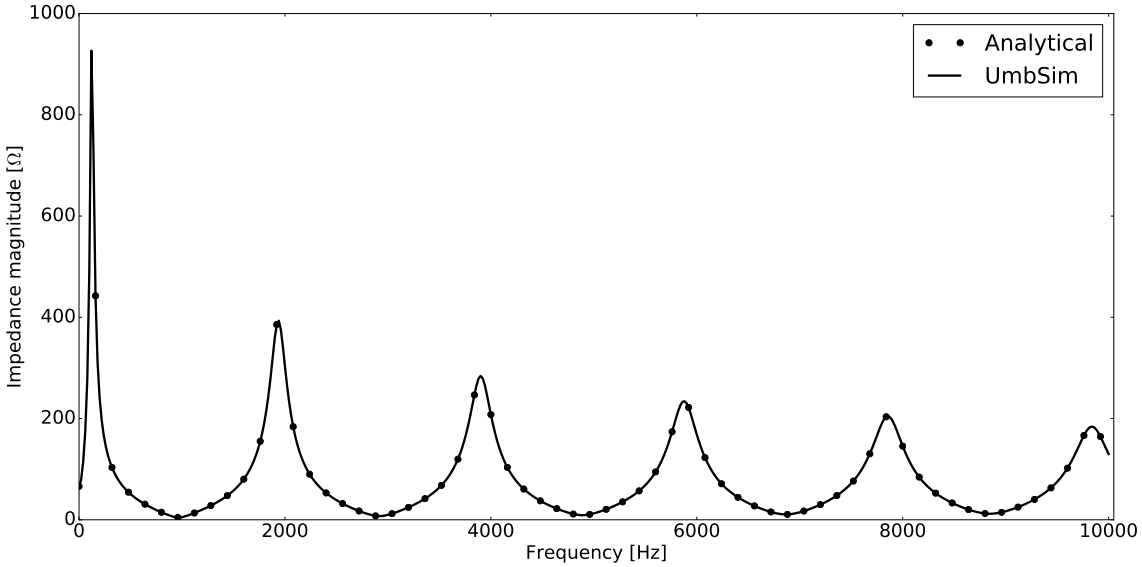


Figure 5.9: Comparison between UmbSim and analytical methods of input impedance magnitude spectrum for Umbilical A1 with nominal load.

termined coefficients i^+ and i^- are found as

$$i^+ = \frac{1}{2} i_L e^{\gamma \boldsymbol{\ell}} \left(\frac{Z_L}{Z_C} + 1 \right) \tag{5.23}$$

$$i^- = \frac{1}{2} i_L e^{-\gamma \boldsymbol{\ell}} \left(\frac{Z_L}{Z_C} - 1 \right) \tag{5.24}$$

Substituting these into Eq. (5.20) yields

$$Z_{in}(\omega) = Z_C \frac{(e^{\gamma \boldsymbol{\ell}} + e^{-\gamma \boldsymbol{\ell}}) \frac{Z_L}{Z_C} + (e^{\gamma \boldsymbol{\ell}} - e^{-\gamma \boldsymbol{\ell}})}{(e^{\gamma \boldsymbol{\ell}} - e^{-\gamma \boldsymbol{\ell}}) \frac{Z_L}{Z_C} + (e^{\gamma \boldsymbol{\ell}} + e^{-\gamma \boldsymbol{\ell}})} \tag{5.25}$$

$$= Z_C \frac{\cosh(\gamma \boldsymbol{\ell}) Z_L + \sinh(\gamma \boldsymbol{\ell}) Z_C}{\sinh(\gamma \boldsymbol{\ell}) Z_L + \cosh(\gamma \boldsymbol{\ell}) Z_C} \tag{5.26}$$

$$= Z_C \frac{Z_L + Z_C \tanh(\gamma \boldsymbol{\ell})}{Z_C + Z_L \tanh(\gamma \boldsymbol{\ell})} \tag{5.27}$$

by recognizing the hyperbolic functions. By plotting Eq. (5.27) along with, e.g. Fig. 5.5a, the plot of the input impedance magnitude in Fig. 5.9 is produced. They are in good agreement, as again is to be expected due to UmbSim being an implementation of the exact solution.

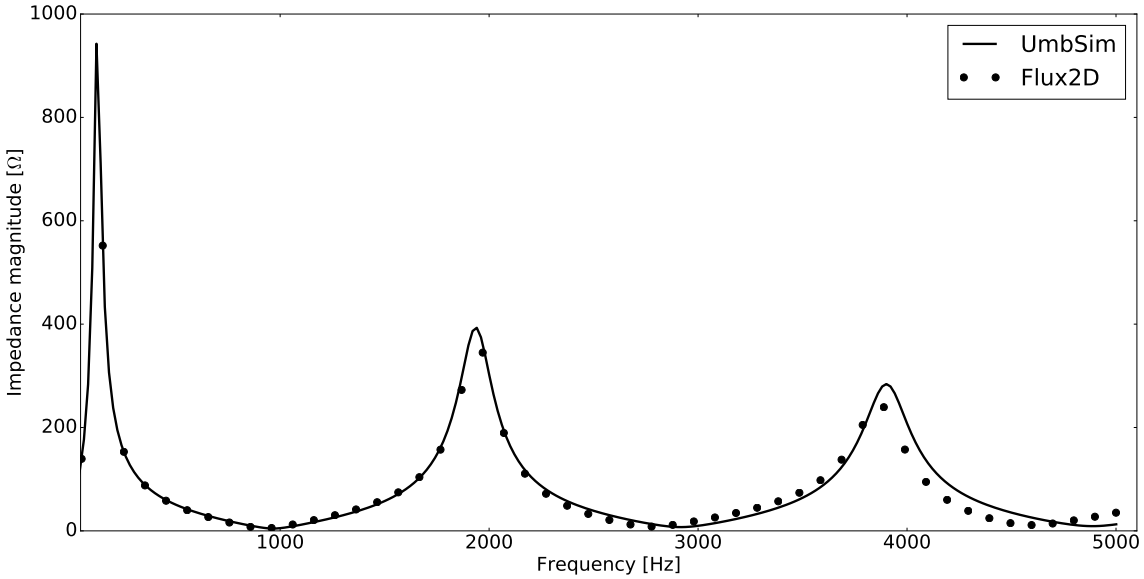


Figure 5.10: Input impedance magnitude spectrum of Umbilical A1 with nominal load. Comparison beetween UmbSim and Flux2D simulations.

Comparison with Flux2D

The comparison between UmbSim and analytical methods in the preceding section serves the purpose of validating the implementation of the `solver.py` and `simulation.py` modules. Since, however, both UmbSim and the analytical approach use the impedance- and admittance matrices computed in the `system.py` module, it does not verify the implementation, nor the precision, of the power umbilical impedance- and admittance formulation.

For this reason, UmbSim will also be compared with simulations ran in Flux2D. The input impedance obtained from UmbSim and Flux2D are plotted together in Fig. 5.10, for Umbilical A1 with nominal load. As can be seen, UmbSim and Flux2D corresponds well for frequencies $< \sim$2.5 kHz. After this, deviations can be seen to increase with frequency. This is thought to be due to the underlying 10 nominal pi model implemented in Flux2D, which does not capture longitudinal effects precisely. This deviation is more prominent for longer cables and higher frequencies.

Additional comparisons with Flux2D will be done in upcoming example problems.

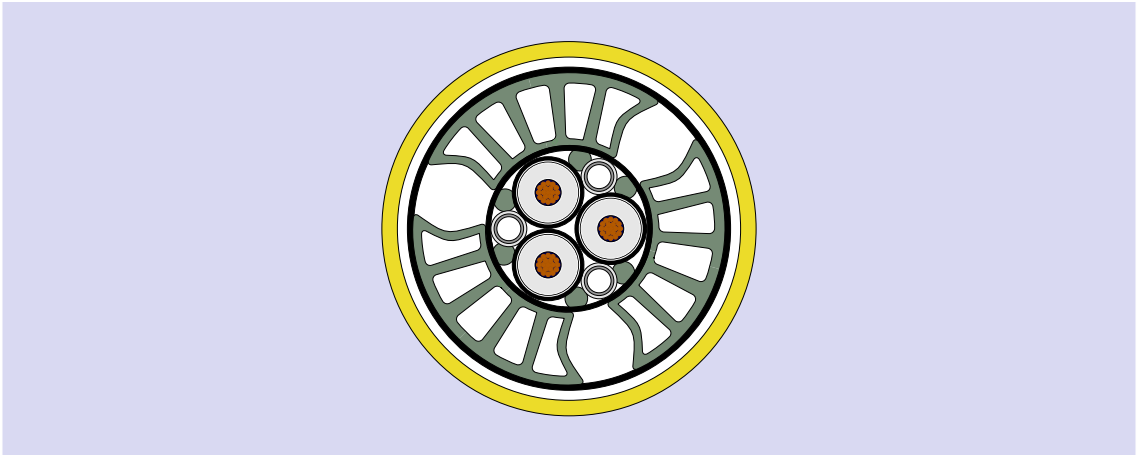


Figure 5.11: Umbilical A2 surrounded by infinite seawater.

5.1.2 Example A2

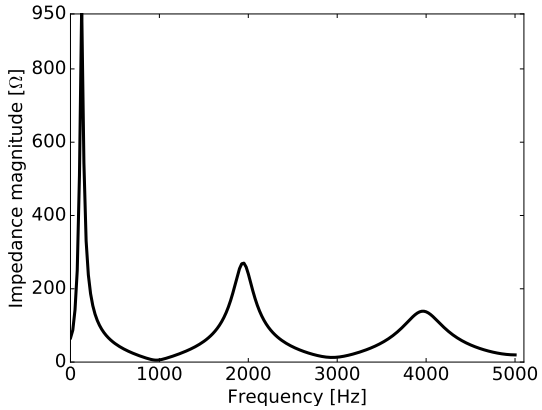
In the next example, the power umbilical shown in Fig. 5.11 will be studied, referred to as Umbilical A2. Notice that the fiber optic cable present in Umbilical A has been replaced with another steel duplex tube. The reason for this is to retain symmetry in the power umbilical, which means it is still sufficient to analyze it per phase. Also, as before, the power umbilical is modelled as armourless.

The source- and load end terminations are the same as for Umbilical A, and are given in Table 5.4.

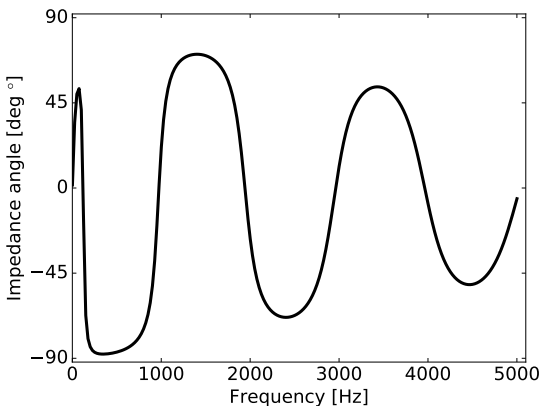
UmbSim simulations

Umbilical A2 along with terminations are defined in a Python script, which can be found in Appendix C.2. Figure 5.12 show simulated input impedance spectra, while Fig. 5.13 show voltages- and currents along power phases at 50 Hz, respectively.

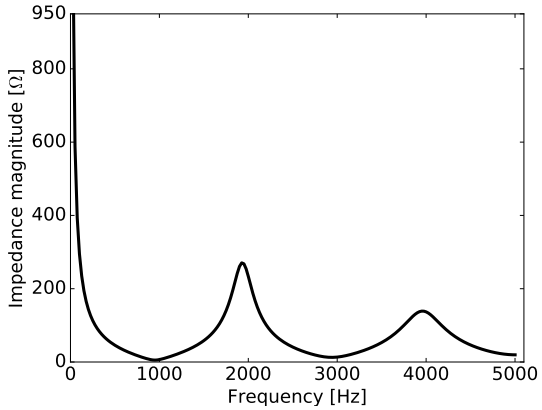
As earlier, resonance phenomena are observed in the impedance spectra, but the parallel resonance peaks for Umbilical A2 are more heavily damped than for Umbilical A1.



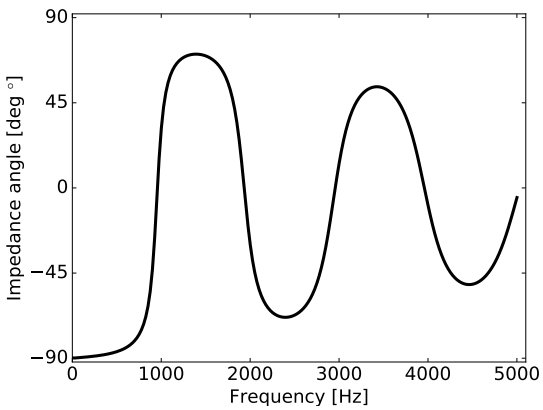
(a) Magnitude, nominal load.



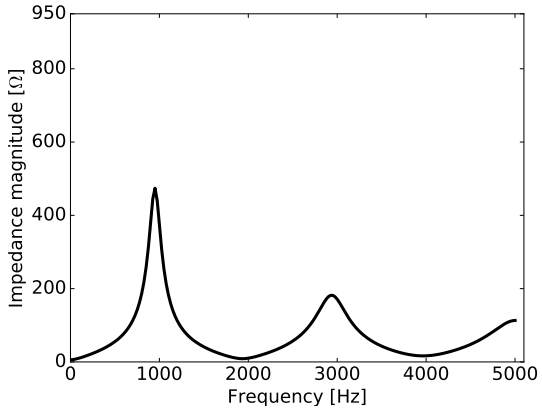
(b) Angle, nominal load.



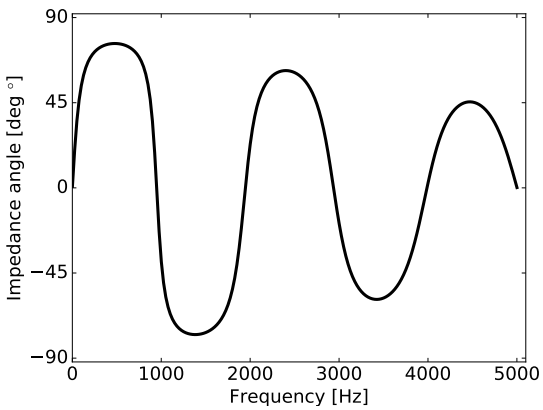
(c) Magnitude, open load end.



(d) Angle, open load end.

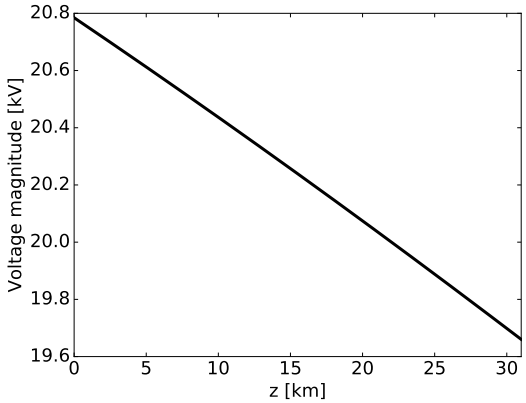


(e) Magnitude, shorted load end.

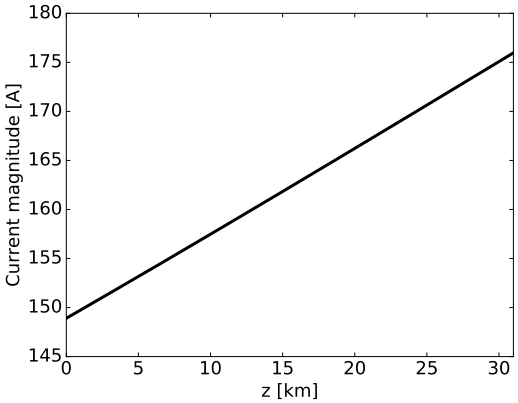


(f) Angle, shorted load end.

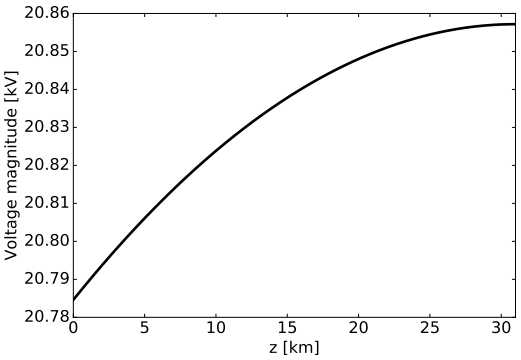
Figure 5.12: UmbSim simulations of input impedance spectra for Umbilical A2 power phases with different terminations.



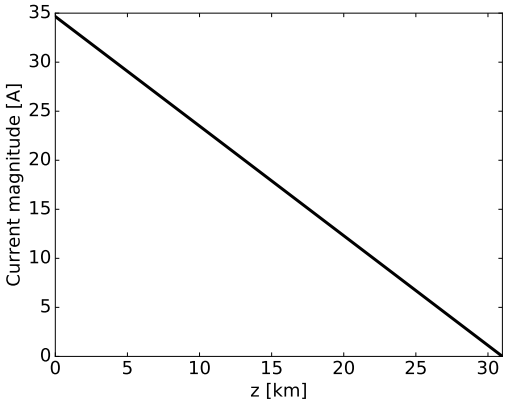
(a) Phase voltage, nominal load.



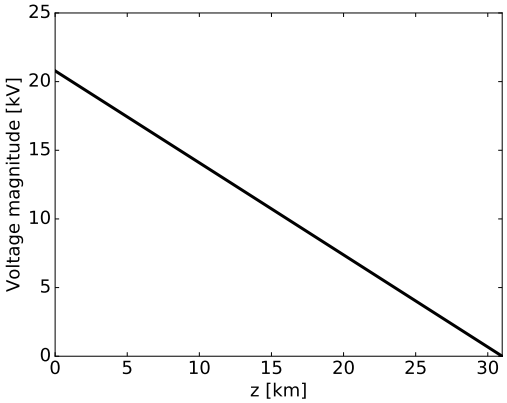
(b) Phase current, nominal load.



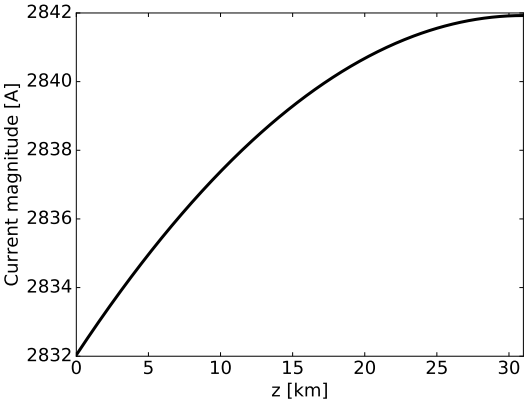
(c) Phase voltage, open load end.



(d) Phase current, open load end.



(e) Phase voltage, shorted load end.



(f) Phase current, shorted load end.

Figure 5.13: UmbSim simulation of voltage- and current magnitudes along power phases at 50 Hz for Umbilical A2.

Table 5.7: Per unit length parameters of Umbilical A2 power phases at 50 Hz.

Parameter	Value
resistance	0.194 Ω/km
inductance	0.431 mH/km
capacitance	0.171 μF/km

Run time

To see what parts of UmbSim is the most time consuming, the script for Example A2 was profiled (without the code related to plotting) [1]. This resulted in a total run time of 4.418 s, with the most time consuming methods from the profiling given in Table 5.8. Together, they comprise over 70 % of the total run time. This is due to

1. Their governing expressions, Eqs. (3.16) and (3.15) are complicated, involving partial sums of infinite series of complicated expressions

2. Their call count is large; 3000 for `zpm()` and 1200 for `zpinner()`

For the first reason, the number of terms included in the partial sum could affect computation time drastically. By inspection it seems that they converge the slowest for elements that are close to the surroundings inner surface. The mutual impedance also seems to converge the slowest when there is a 45° angle between two elements. By plotting these to equations as a function of the number of terms n included in the partials sums, Figs. 5.14a and b are produced. As default in UmbSim, n is set to 25, which might be unnecessary high. The rate of convergence could change, however, if the surrounding's parameters change.

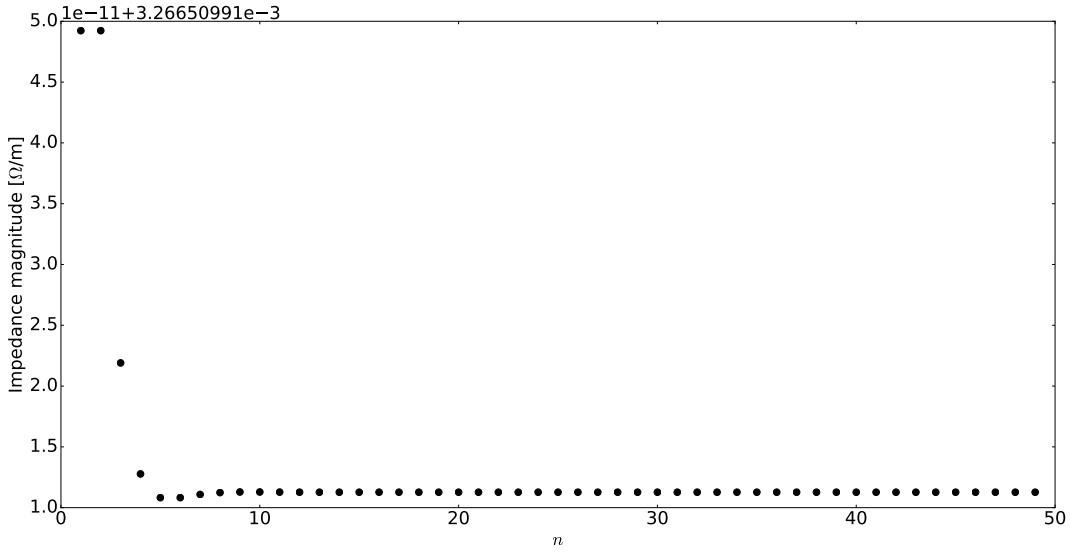
As for the call counts, they can grow rather large, since the `zpm()` is called for each mutual coupling between conductors. The number of calls can be found by considering a power umbilical with N conductors. For each conductor to have mutual couplings with all other conductors requires it to have $N-1$ mutual couplings. Since there are N conductors the number of mutual couplings are $N(N-1)$, but the impedance matrix is symmetric, and so the number of times the `zpm()` method has to be called is

$$\text{Call count } \texttt{zpm()} = \frac{N(N-1)}{2} \tag{5.28}$$

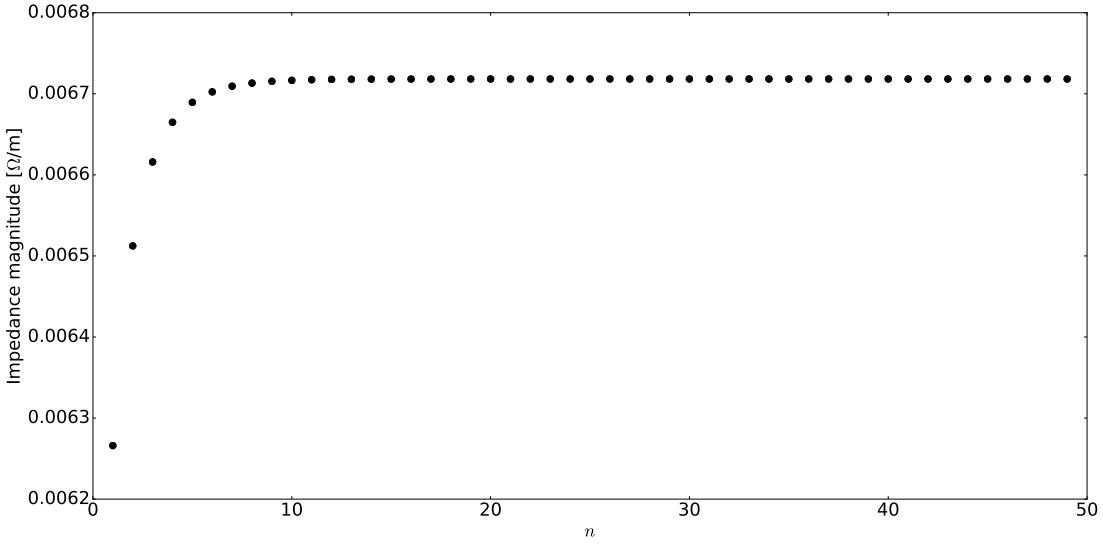
For 6 conductors as was the case with Example A2, `zpm()` had to be called 15 times for each frequency. If, for example, one were to model the outer layer of the power umbilical depicted in Fig. 2.5, it would be a total of 22 conductors (counting electrical quads), which means that the number of times `zpm()` would have to be called is 231 per frequency, or 46200 times for 200 frequencies.

In some texts, there have been some concern to, by the author's impressions, computation time due to the transformation matrix $\mathbf{T}_I$ and the diagonal matrix Λ^2 being frequency dependent, which means $\mathbf{YZ}$ have to be diagonalized for each frequency (Paul 2008; F. F. d. Silva and Bak 2013). As can be seen from Table 5.8, the diagonalization is done through NumPy's `linalg.eig()` method, which returns the eigenvalues and -vectors of a matrix. It is called exactly 200 times - the number of frequencies simulated, but it takes very little time compared to the methods discussed above - only 48 ms out of a total simulation time of 4418 ms.

[1] The simulations are ran on a low-to-medium performance laptop by 2017 standard, with an Intel Core i5-5200 CPU and 8 GB RAM



(a) Mutual impedance.



(b) Impedance of the surrounding's inner surface

Figure 5.14: Eqs. (3.16) and (3.15) plotted with increasing number of terms n in the partial sums.

Method	Description (Package)	Call count	Time [ms]	% of total run time
`system.zpm()`	Mutual impedance	3000	2335	52.9
`system.zpinner()`	Impedance of pipe's inner surface	1200	814	18.4
`system.impedance_matrix()`	Assemble impedance matrix	200	82	1.9
`<sum>`	Summing lists (built-in)	4200	57	1.3
`<numpy.core.multiarray.array>`	Array initiation (NumPy)	10703	47	1.1
`eig`	Eigenvalues and -vectors (NumPy)	200	45	1.0
`<math.cos>`	Cosine (math)	75000	43	1.0
`inv`	Matrix inversion (NumPy)	401	39	0.9
Total run time: 4418 ms, Number of frequencies simulated: 200				

Table 5.8: Profiling for the script used in Example A2.

Comparison with analytical calculations

To test UmbSim's handling of steel duplex tubes, an analytical formula where the duplex steel voltage is calculated for low frequencies is derived.

Duplex voltage at low frequencies Assume that the induced current in a steel duplex tube is only due to the three power phases. That is, the contribution from currents flowing in the other two steel duplex tubes is neglible. Then the voltage on the steel duplex tube, with subscript 1, is given by the first telegrapher's equation

$$-\frac{dv_1(z)}{dz} = (r_{11} + j\omega l_{11})i_1(z) + j\omega \sum_{k=2}^{4} l_{1k} i_k(z) \tag{5.29}$$

where l_{1k} is the mutual inductance between the steel duplex tube and $i_k(z)$ is the current in the k-th power phase, respectively. Likewise, the change in current in the steel duplex tube is governed by the second telegrapher's equation

$$-\frac{di_1(z)}{dz} = j\omega c_1 v_1(z) \tag{5.30}$$

Taking the derivative of Eq. (5.29) yields

$$-\frac{d^2v_1(z)}{dz^2} = r_{11}\frac{di_1}{dz} + j\omega \sum_{k=1}^{4} l_{1k}\frac{di_k(z)}{dz} \tag{5.31}$$

where it is assumed that the power umbilical is uniform along its length. For low frequencies, the induced voltage in the duplex is low, so that the current can be assumed constant with respect to z. Hence,

$$-\frac{d^2v_1(z)}{dz^2} = j\omega \sum_{k=2}^{4} l_{1k}\frac{di_k(z)}{dz} \tag{5.32}$$

The current derivative on these power phases is described by the second telegrapher's equation as

$$-\frac{di_k(z)}{dz} = j\omega c v_k(z) \tag{5.33}$$

for all three phases, since the power phases have equal capacitance. Inserting Eq. (5.33) into (5.32) gives

$$-\frac{d^2v_1(z)}{dz^2} = \omega^2 c \sum_{k=2}^{4} l_{1k} v_k(z) \tag{5.34}$$

Since two of the conductors have the same distance to the duplex and hence the same inductance, and using the fact that the voltages are in positive sequence gives

$$-\frac{d^2v_1}{dz^2} = \omega^2 c(l_{12} + a^2 l_{13} + a l_{13})v(z) \tag{5.35}$$

where $v(z)$ is the magnitude of the phase voltage and $a = e^{j\frac{2\pi}{3}}$. Since $a^2 + a = -1$,

$$-\frac{d^2 v_1}{dz^2} = \omega^2 c(l_{12} - l_{13})v(z) \tag{5.36}$$

where $v(z)$ needs to be determined. For low frequencies, in the dc limit, the telegrapher's equations for a power phase is written as

$$\frac{dv(z)}{dz} = -ri \tag{5.37}$$

so there is a linear drop in voltage along the power phase. Since the load is purely resistive at dc, the current drawn must be equal to

$$i_s = \frac{v_s}{r\boldsymbol{\ell} + R_L} \tag{5.38}$$

and hence the change in voltage along the power phase is

$$\frac{dv(z)}{dz} = -\frac{r}{r\boldsymbol{\ell} + R_L} v_s \approx -61 \text{ V/km} \tag{5.39}$$

One could solve Eq. (5.37) and use the solution to solve for the steel duplex tube voltage, but since the voltage drop is relatively small across the length of the power phase, a mean value for the power phase voltage is taken

$$v(z) \approx \bar{v} = v(z = \frac{\boldsymbol{\ell}}{2}) = 19.8\text{kV} \tag{5.40}$$

so the general solution of Eq. (5.36) is given as

$$v_1(z) = -\frac{1}{2}\omega^2 c(l_{12} - l_{13})\bar{v}z^2 + C_1 z + C_2 \tag{5.41}$$

where C_1 and C_2 are constants yet to be determined. Since the duplex is grounded at both ends, $v_1(0) = v_1(\boldsymbol{\ell}) = 0$. By imposing the boundary conditions, the specific solution then becomes

$$v_1(z) = -\frac{1}{2}\omega^2 c(l_{12} - l_{13})\bar{v}(z^2 - z\boldsymbol{\ell}) \tag{5.42}$$

which is the equation for a parabola. Plots of the magnitude of the duplex voltage achieved from both the analytical solution and UmbSim are given in Fig. 5.15. The deviation is quite small, which indicates that the assumptions are valid, and that the implementation of steel duplex tubes in UmbSim appears to be correct.

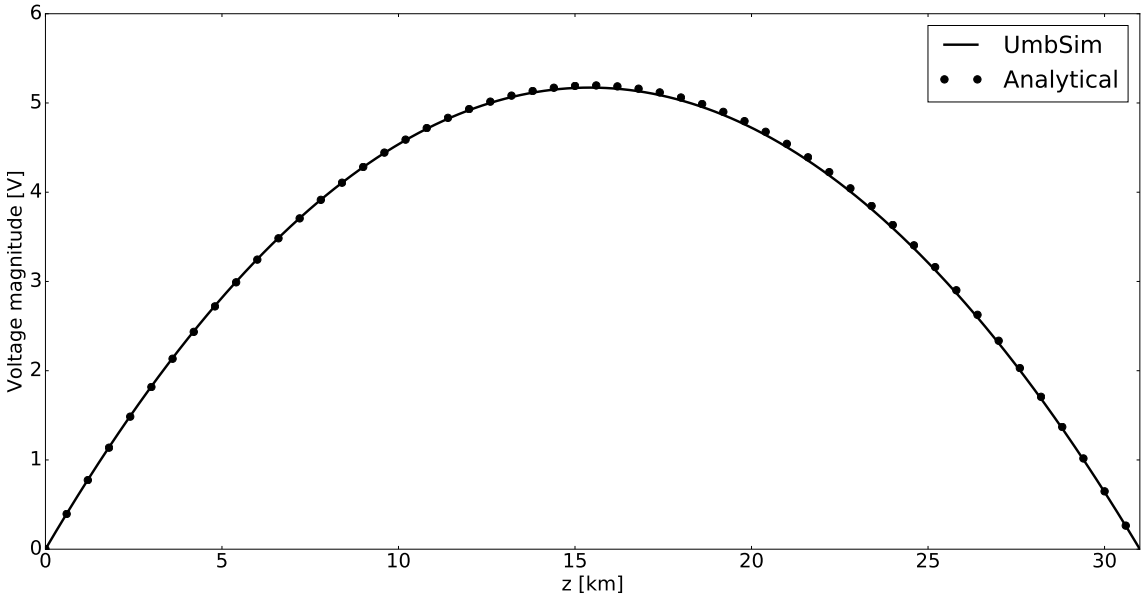


Figure 5.15: rms voltage along one of the steel duplex tubes in Umbilical A2 at 50 Hz. Comparison of analyticalmethods and simulation in UmbSim.

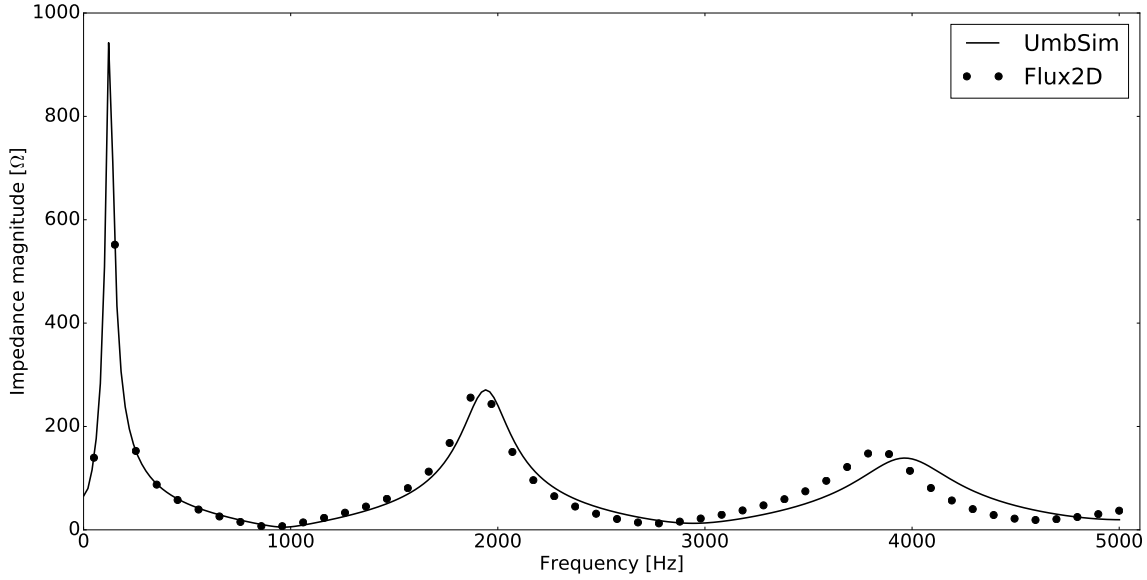


Figure 5.16: Input impedance magnitude spectrum of Umbilical A2 with nominal load.

Comparison with Flux2D

Figure 5.16 shows the input impedance spectrum for Umbilical A2, from both UmbSim and Flux2D simulations. As earlier, they coincide well for low frequencies, but for higher frequencies there is a horizontal shift of the impedance curves. This is again thought to be due to the inaccuracies of the 10 nominal pi model implemented in Flux2D.

To check if the deviation between UmbSim and Flux2D is due to differences in the calculated per unit length parameter values, a very short version (1 m) of Umbilical A2 were modelled in both programs. A short cable means that the capacitance is negligible, and the per unit length resistance and inductance can be found from the real- and imaginary part of the input impedance, respectively. Plots for the per unit length resistance and inductance are given in Fig. 5.17a and 5.17b, respectively. The difference in r and l is expected, as Flux2D includes proximity effects, which would effectively yield an increase in resistance and a decrease in inductance. However, this does not explain the observed shift in the impedance spectra, as will become clear in the following paragraphs.

Since resonance frequencies occur at values proportional to $\frac{1}{\sqrt{lc}}$, and c is equal for both programs (0.17 μF/km), only the difference in inductance values would yield a shift in frequency. However, since the per unit length inductance is *higher* for UmbSim than for Flux2D, it should mean that the resonance frequencies of UmbSim should be shifted to the left of the resonance frequencies of Flux2D in Fig. 5.16. This is clearly not true, since UmbSim is actually shifted to the right of Flux2D.

Therefore, since

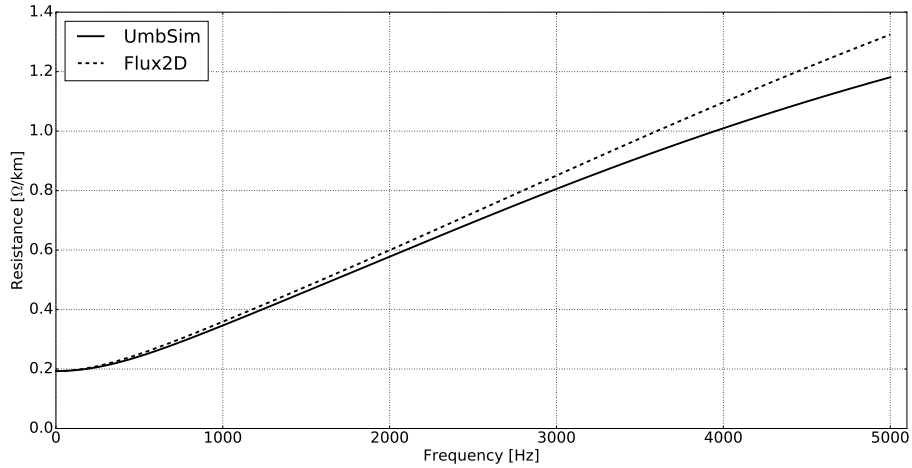
1. The `solver.py` and `simulation.py` modules are validated against analytical methods with practically negligible errors

   and

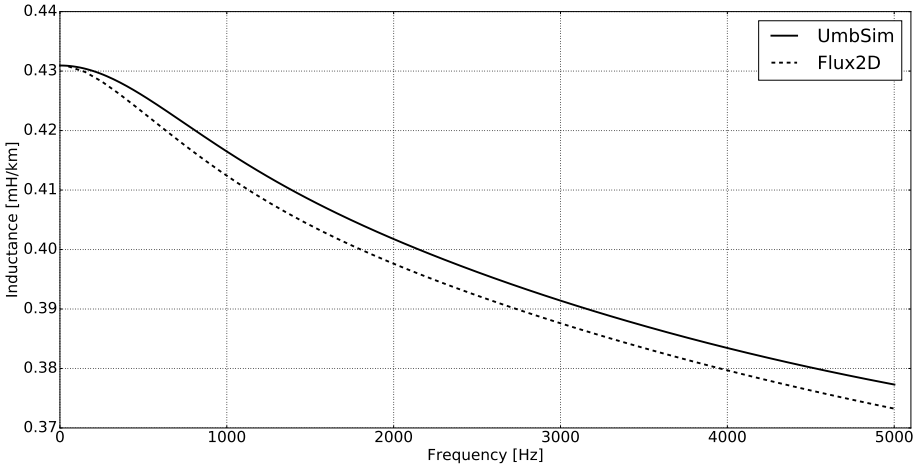
2. The expected shift between impedance curves due to differences in per unit length parameters between UmbSim and Flux2D contradicts the observed shift

it is concluded that the shift in frequency is due to the weakness in the ability of the 10 nominal pi model implemented in Flux2D to capture longitudinal effects beyond the first series resonance.

To verify this, a corrected input impedance spectrum from Flux2D can be produced. By using the exact formula for the input impedance, namely Eq. (5.27), along with the Flux2D per unit length resistance and inductance values, the plot in Fig. 5.18 is produced. The errors are now solely due to proximity, and are also smaller than before the correction. The left horizontal shift of the UmbSim curve with respect to Flux2D, and a slight dampening of the Flux2D parallel impedance resonance peaks, fits well with what one would expect from proximity effects.



(a) Per unit length resistance.



(b) Per unit length inductance.

Figure 5.17: Comparison of UmbSim and Flux2D simulations of per phase per unit length resistance and inductance for Umbilical A2.

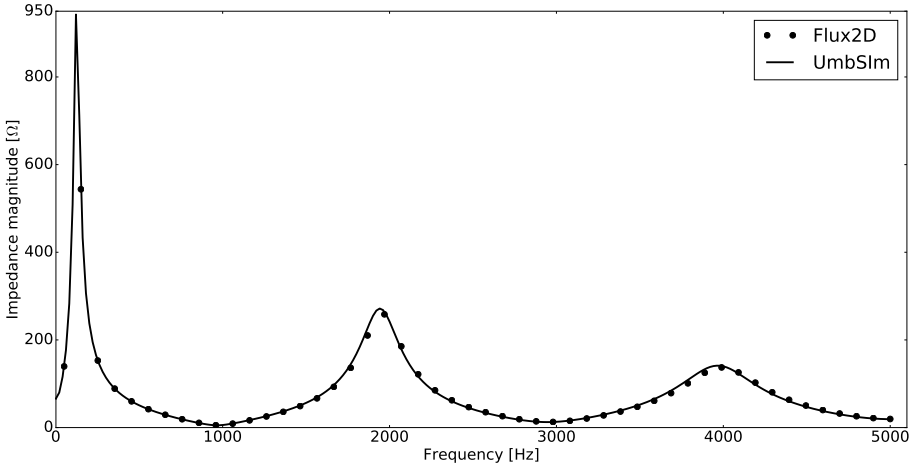


Figure 5.18: Comparison between UmbSim and corrected Flux2D input impedance spectra.

5.1.3 Example A3 - Harmonic analysis

The International Electrotechnical Commission (IEC) standard 61000-2-4 gives recommendations for the maximum acceptable amount of harmonic content in a point of common coupling (PCC) in a power system. The standard is further divided into classes, in which a subsea power system can be regarded as belonging to Class 2 (Norsk Elektroteknisk Komite 2002). In Table 5.9 the maximum recommended values of the harmonic voltages of order h are given as percentages of the fundamental. Figure 5.19 gives a visualization of this spectrum.

In addition to maximum limits for each harmonic, the standard also gives recommendations for the maximum *total harmonic distortion* (THD) for the voltage. Voltage THD is a measure of the amount of harmonic content in the system voltage and is given as

$$THD_V = \sqrt{\sum_{h=2}^{50} (\frac{v_h}{v_1})^2} \cdot 100\% \tag{5.43}$$

where v_h is the magnitude of the harmonic voltage of order h (Mohan, Undeland, and Robbins 2003). IEC 61000-2-4 recommends the voltage THD at a PCC to be no more than 8 %, but if the system voltage had harmonic content according to Table 5.9, the THD would be 11.55 %, which exceeds the maximum limit.

An important aim for this thesis is that one should be able to study the effect harmonic content would have on a system. Built-in routines are not included as of now, but scripts can be written manually to analyze such situations. Although an unrealistic scenario, it would be interesting to study the response of Umbilical A2 if a distorted voltage defined by the values in Table 5.9 were applied to the source end. This can be done by the principle of superposition, by looping over each harmonic component (with sending end voltage magnitude as defined in IEC 61000-2-4). Then, finding the systems response to each harmonic (rms currents and voltages) and in the end computing the sums

$$v_k(z) = \sqrt{v_1^2(z) + \sum_{h=2}^{50} v_h^2(z)} \tag{5.44}$$

$$i_k(z) = \sqrt{i_1^2(z) + \sum_{h=2}^{50} i_h^2(z)} \tag{5.45}$$

would yield the rms voltage and current in the k-th conductor in the power umbilical in response to every applied harmonic.

It is important to know the sequence of the harmonic voltage sources. For all voltage harmonics, the magnitude of each phase voltage source v_{sh} is equal, but the direction of rotation or the phase shift between them will vary. For example, for the 1^{st} harmonic, the voltage source is in positive sequence as usual, i.e.

$$\mathbf{v}_{s1} = (v_{s1},\ v_{s1}a^2,\ v_{s1}a)^t$$

Table 5.9: Harmonic voltage limits at point of common coupling as recommended by IEC 61000-2-4 Class 2.

Odd non-triplen harmonics		Triplen harmonics		Even harmonics	
Harmonic order h	v_h [%]	Harmonic order h	v_h [%]	Harmonic order h	v_h [%]
5	6	3	5	2	2
7	5	9	1.5	4	1
11	3.5	15	0.4	6	0.5
13	3	21	0.3	8	0.5
17	2	27 - 45	0.2	10	0.5
19 - 49	x_1	-	-	12 - 50	x_2
$x_1 = 2.27 \cdot (17/h)$ - 0.27					
$x_2 = 0.25 \cdot (10/h) + 0.25$					

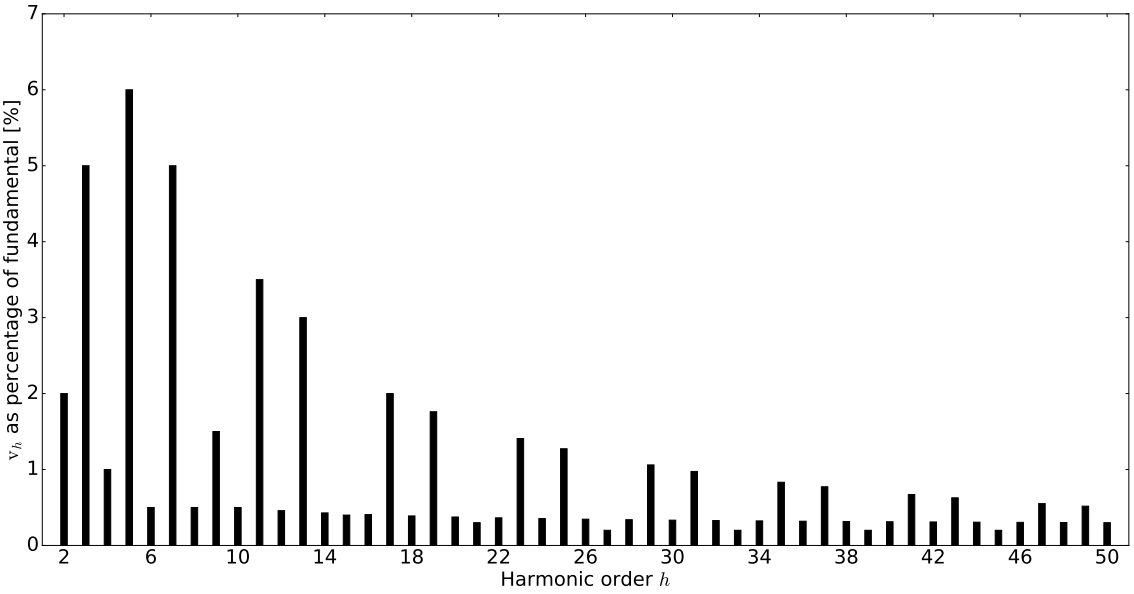


Figure 5.19: Limits of harmonic voltages as defined by IEC 61000-2-4 Class 2.

while for the 2nd harmonic, the voltages is in negative sequence

$$\mathbf{v}_{s2} = (v_{s2},\ v_{s2}a,\ v_{s2}a^2)^t$$

and for the 3rd harmonic, there is no phase shift between the three voltage sources, so the voltage source is written as

$$\mathbf{v}_{s3} = (v_{s3},\ v_{s3},\ v_{s3})^t$$

and then the 4th harmonic is in positive sequence, the 5th in negative sequence, the 6th in zero sequence, and so on. Therefore, for every harmonic h the voltage source vector with correct sequence can be represented by

$$\mathbf{v}_{sh} = v_{sh}(1,\ a^{2h},\ a^h)^t \tag{5.46}$$

since $a^3 = 1$.

It is also important to describe the load correctly for the three sequences. For this example problem, the loads seen by the positive- and negative sequence loads are assumed equal, although this might not be the case for rotating machines. The zero sequence, however, will see an open-circuited load end, as would be the case for a step-down transformer with an isolated neutral. Therefore, the per phase loads are given as

$$\begin{aligned} R_L^+ &= R_L^- = 60\ \Omega \\ L_L^+ &= L_L^- = 0.3\ \text{H} \\ R_L^0 &= 10^5\ \Omega \\ L_L^0 &= 0 \end{aligned}$$

where superscripts $+$, $-$ and 0 denote positive, negative and zero sequence impedance, respectively.

UmbSim simulation

The harmonic analysis on Umbilical A2 is performed by running the script in Appendix C.3. Umbilical A2 is defined as in the preceding example, but code for modelling the above scenario with inclusion of voltage harmonics is written as well.

The simulation produces the rms voltages and currents along the length of Umbilical A2 for both power phases and steel duplex tubes, which can be seen in the plots of Figs. 5.20 to 5.23.

As is to be expected, the harmonic content effectively leads to a voltage rise on the power phase. The difference between the distorted voltage and the fundamental increases along the length of the cable, from around 150 volts at the sending end to a difference of around 600 volts at the receiving end.

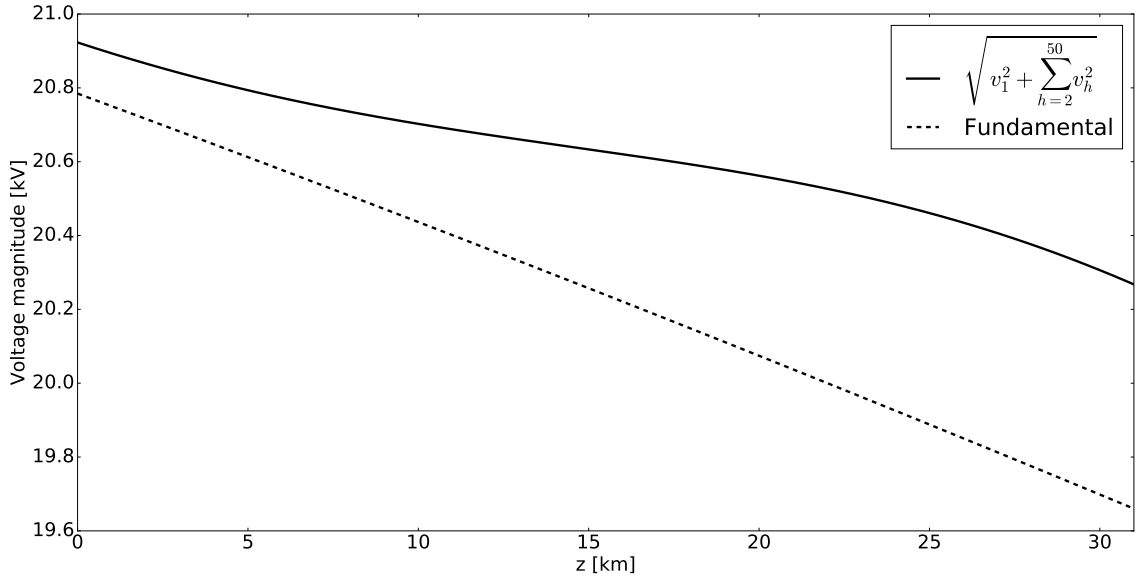


Figure 5.20: rms voltage on a power phase along Umbilical A2 when a harmonic voltage spectrum as defined by the limits in IEC 61000-2-4 Class 2 is applied at the source end.

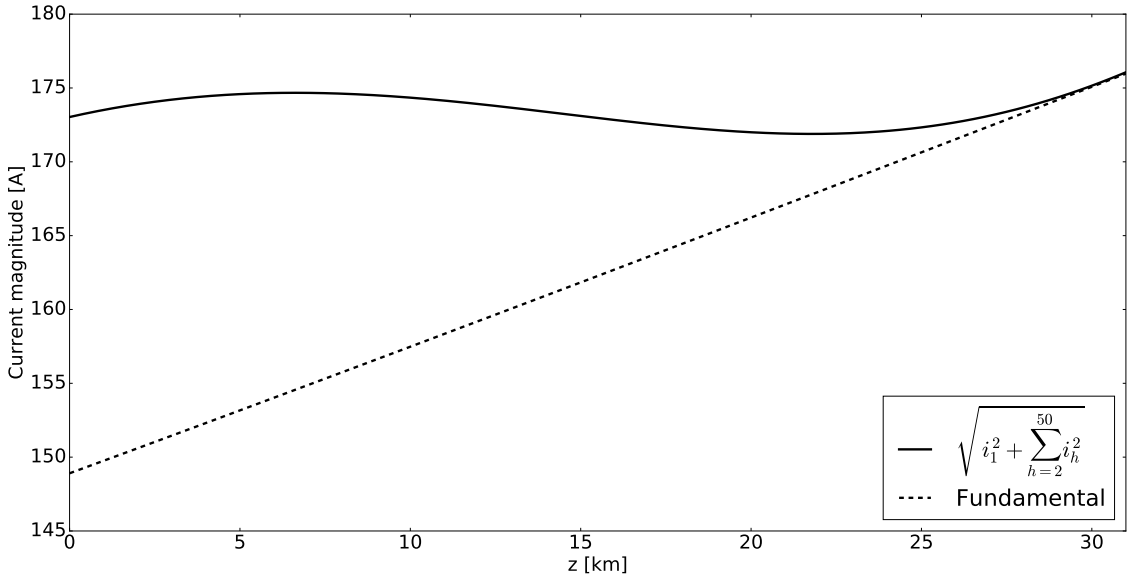


Figure 5.21: rms current in a power phase along Umbilical A2 when a harmonic voltage spectrum as defined by the limits in IEC 61000-2-4 Class 2 is applied at the source end.

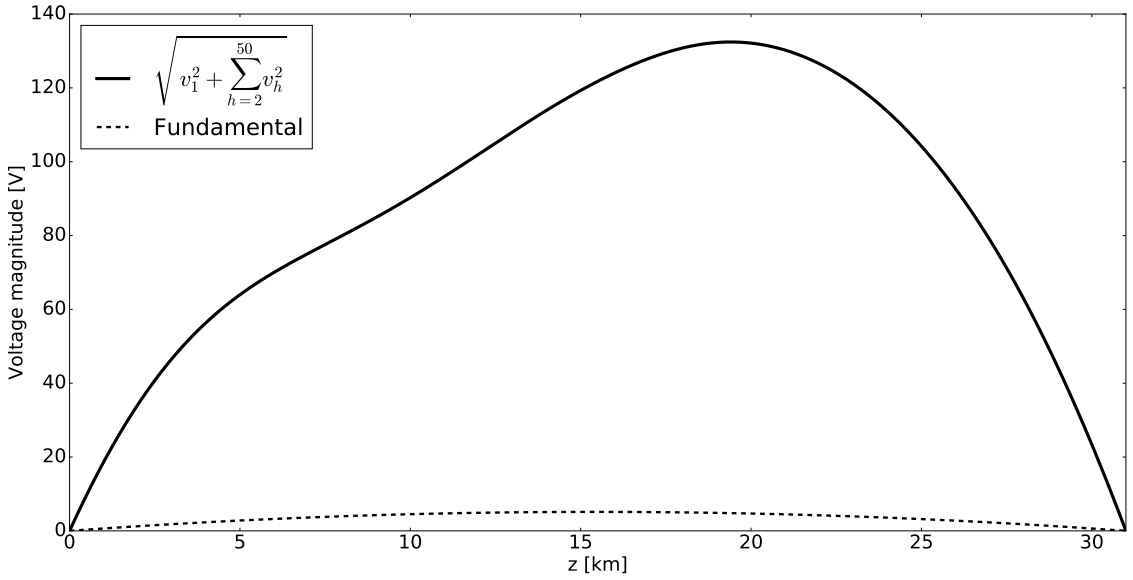


Figure 5.22: rms voltage on a steel duplex tube along Umbilical A2 when a harmonic voltage spectrum as defined by the limits in IEC 61000-2-4 Class 2 is applied at the source end.

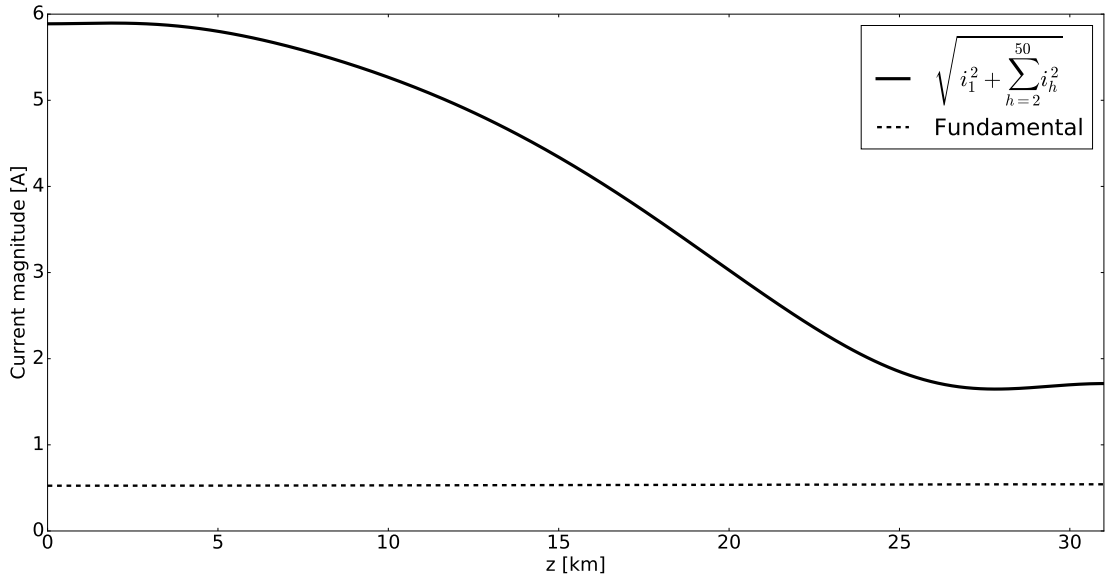


Figure 5.23: rms current in a steel duplex tube along Umbilical A2 when a harmonic voltage spectrum as defined by the limits in IEC 61000-2-4 Class 2 is applied at the source end.

Comparison with analytical calculations

Due to the number of terms involved in the harmonic analysis, there are not many analytical exercises that are realistic to perform. Nevertheless, a simple check related to the distortion of the voltage can be calculated.

From the IEC 61000-2-4 Class 2 spectrum, the THD_V of the source is found to be 11.55 %. The source voltage should have a magnitude equal to

$$v_s = \sqrt{v_{s1}^2 + \sum_{h=2}^{50} v_{sh}^2} = v_{s1}\sqrt{1 + \sum_{h=2}^{50} \frac{v_{sh}^2}{v_{s1}^2}} = v_{s1}\sqrt{1 + THD_V^2} = 1.0066 v_{s1}$$

which coincides well with UmbSim, which gives an output of 20.923 kV at $z = 0$, which is an increase of 0.666 % relative to the fundamental source voltage.

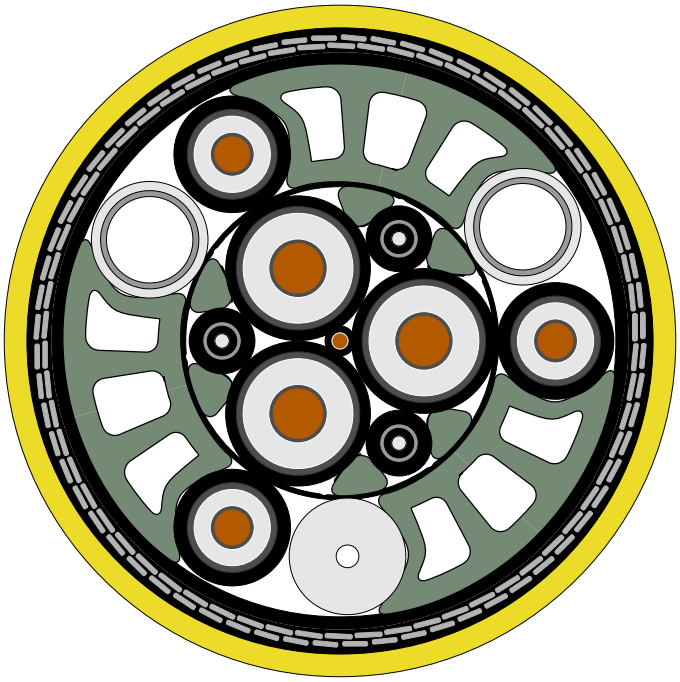


Figure 5.24: Umbilical B.

5.2 Case study - Umbilical B

The second case study that will be modelled in UmbSim is the 42 km long power umbilical in Fig. 5.24, for ease of reference denoted Umbilical B.

Its inner layer consists of a 24 kV three-phase power phase circuit with 95 mm^2 conductors. It also has three fiber optic elements and a neutral wire placed at the center of the power umbilical. As for Umbilical A, the fiber optic elements are neglected, but data for the inner circuit power phases are found in Table 5.10.

The outer layer consists of a 12 kV three-phase power phase circuit with 50 mm^2 conductors. It also has two duplex steel tubes placed next to the upper two power phases. Adjacent to the last power phase is a HDPE bolt, which has no conducting parts. Data for the outer circuit power phases and for the steel duplex tubes are found in Table 5.11 and 5.12, respectively.

Both the power phases of the inner- and outer circuit have solid conductors and are without metallic screens. Umbilical B also have HDPE filler elements (in green) and stranded steel wires as armour. Some general data for Umbilical B and its surroundings are given in Table 5.15.

As earlier, if μ and/or ϵ are not given for a specific parts of the power umbilical, their value is either taken as their respective values in free space, or they are irrelevant when modelling.

The measurements on Umbilical B are performed separately on the inner- and outer circuit. When measuring on one circuit, its sending end is excited by a positive sequence voltage source with some magnitude. The other circuit is then left open in both ends. After the measurements are conducted, the terminations are reversed. The terminations for both situations are given in Tables 5.13 and 5.14

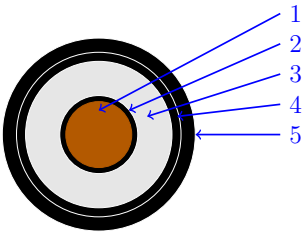


Figure 5.25: One of the power phases in the inner layer of Umbilical B.

Table 5.10: Data for the inner layer power phases in Umbilical B.

	Power phase	Diameter [mm]
1	conductor, 95 mm^2 solid Cu	10.9
2	semi-conducting screen, XLPE	12.9
3	insulation, XLPE	24.2
4	semi-conducting screen, XLPE	27.2
5	semi-conducting polyethylene	31.9
	Electrical properties	Value
	ϵ_r of XLPE	2.4
	dc resistance of conductor (20 °C)	0.193 Ω/km

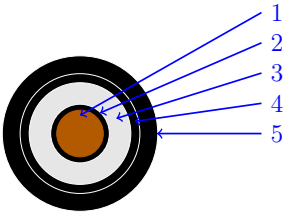


Figure 5.26: One of the power phases in the outer layer of Umbilical B.

Table 5.11: Data for the outer layer power phases in Umbilical B.

	Power phase	Diameter [mm]
1	conductor, 50 mm^2 solid Cu	7.7
2	semi-conducting screen, XLPE	9.7
3	insulation, XLPE	16.8
4	semi-conducting screen, XLPE	19.8
5	semi-conducting polyethylene	25.7
	Electrical properties	Value
	ϵ_r of XLPE	2.4
	dc resistance of conductor (20 °C)	0.387 Ω/km

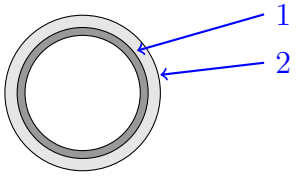


Figure 5.27: One of the steel duplex tubes in Umbilical B.

Table 5.12: Data for the steel duplex tubes in Umbilical B.

	Duplex	Diameter [mm]
1	steel duplex tube, ID=19.05 mm	21.65
2	HDPE sheath	25.7
	Electrical properties	Value
	ϵ_r of HDPE	2.3
	μ_r of duplex steel	32
	Resistivity of duplex steel (20 °C)	$8 \cdot 10^{-7}$ Ωm

Table 5.13: Terminations for the elements in Umbilical B when measuring on the outer circuit.

Source end	
Outer circuit power phase line-to-line voltage	12 kV
Frequency	0 - 5 kHz
Phase sequence	positive
Inner circuit power phases	open
Steel duplex tubes	open
R_S for outer circuit power phases	0
L_S for outer circuit power phases	0
Load end (nominal)	
Outer circuit power phases	open
Inner circuit power phases	open
Steel duplex tubes	open

Table 5.14: Terminations for the elements in Umbilical B when measuring on the inner circuit.

Source end	
Inner circuit power phase line-to-line voltage	24 kV
Frequency	0 - 5 kHz
Phase sequence	positive
Outer circuit power phases	open
Steel duplex tubes	open
R_S for inner circuit power phases	0
L_S for inner circuit power phases	0
Load end (nominal)	
Outer circuit power phases	open
Inner circuit power phases	open
Steel duplex tubes	open

Table 5.15: Various data for Umbilical B and surroundings.

Umbilical B	
Length	42 km
Outer diameter	148 mm
μ inside umbilical	μ_0
Surroundings	
μ air	μ_0
ρ air	$1.3 \cdot 10^{14}$ Ωm

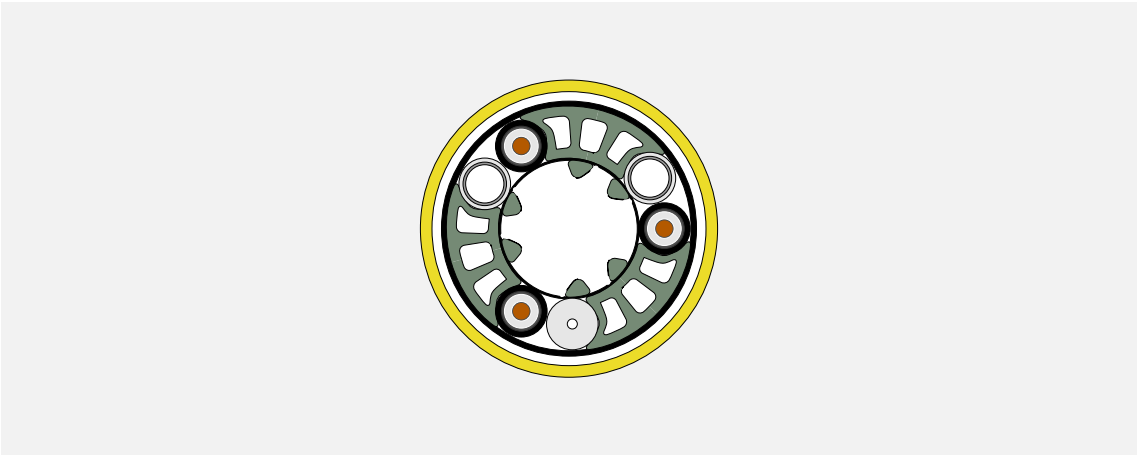


Figure 5.28: Umbilical B1.

5.2.1 Example B1

First, the outer circuit of Umbilical B will be modelled, referred to as Umbilical B1. As explained in Chapter 3, the inner circuit will be neglected when modelling the outer circuit. As earlier, armour will be neglected.

Notice that since Umbilical B1 has an unsymmetrical design with only two steel duplex tubes, all three phases must be studied explicitly to give a full representation of the power umbilical. The three phases will be referred to as phase 1, 2 and 3, respectively, starting from the rightmost power phase and counting counterclockwise.

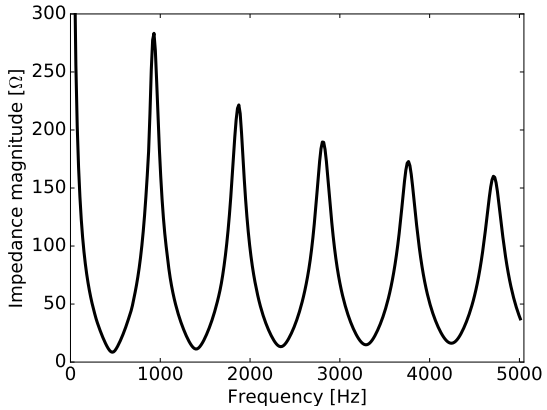
The terminations for Example B1 is given in Table 5.13.

UmbSim simulation

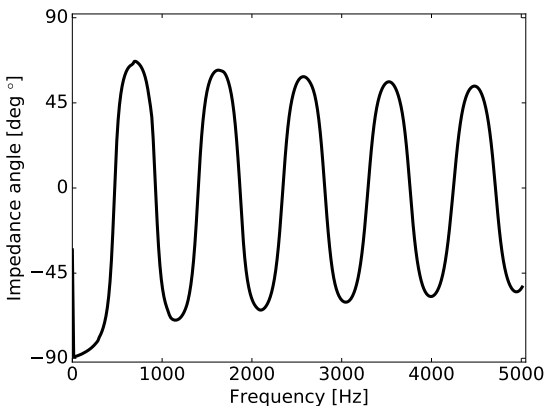
By running the script in Appendix C.4, input impedance spectra are generated for each phase in the outer power circuit. Plots of both magnitude and angle are given in Figs. 5.29a to f.

As can be seen from the plots, none of the input impedance spectra are equal, which is expected.

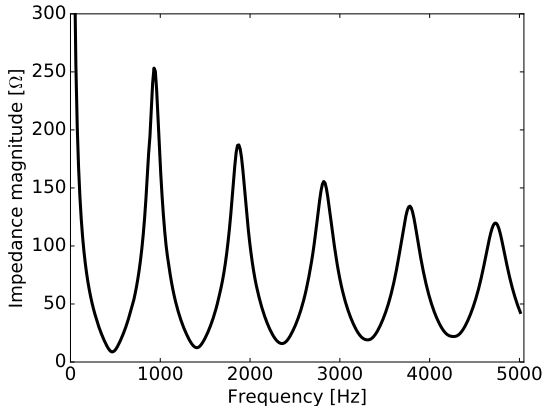
The impedance magnitude spectrum that has the highest maxima is phase 3 - which lies furthest away from duplex steel tubes. Hence, the impact of the steel duplex tubes on phase 3 is small. Phase 1 is slightly more damped than phase 3, and slightly less damped that phase 2. Again, this can be understood from noting that the distance from phase 1 to the steel duplex tube adjacent to phase 2 is larger than the distance between phase 2 and the steel duplex tube adjacent to phase 1.



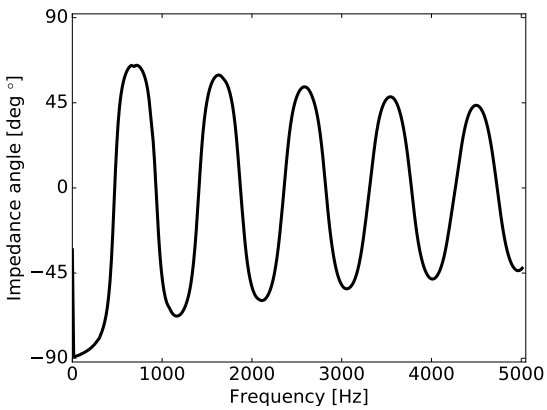
(a) Magnitude, phase 1.



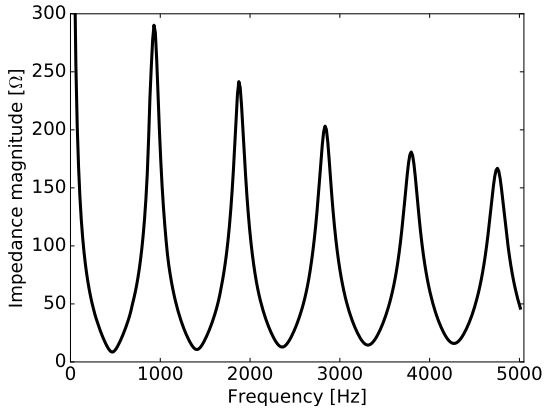
(b) Angle, phase 1.



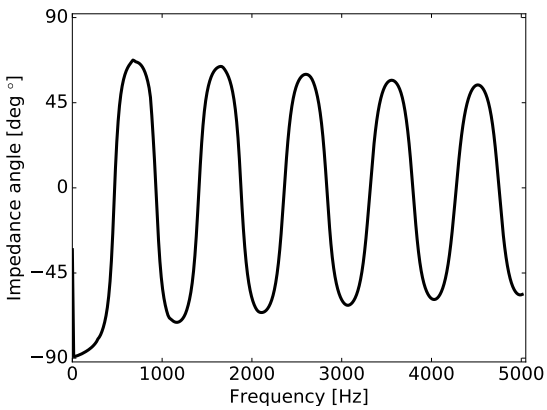
(c) Magnitude, phase 2.



(d) Angle, phase 2.



(e) Magnitude, phase 3.



(f) Angle, phase 3.

Figure 5.29: UmbSim simulation of input impedance spectra for Umbilical A2 power phases.

Comparison with measured values

By conducting a frequency sweep on the power umbilical the input impedance spectra for all three phases can be obtained. The measurements, provided by Nexans, are plotted together with the input impedance magnitudes from UmbSim. The plots can be seen in Fig. 5.30.

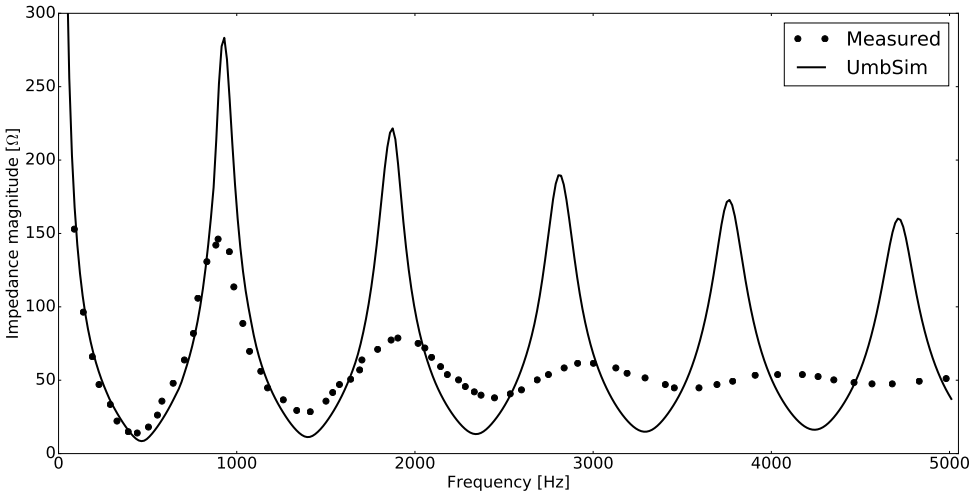
In these plots, the limits of UmbSim become evident. The simulated spectra only fit well with the measured values for frequencies $< \sim 800$ Hz. At higher frequencies they deviate in terms of both magnitude and horizontal alignment.

UmbSim has been validated to handle power phases and steel duplex tubes relatively well in preceding example problems. Therefore, the observed deviation at relatively low frequencies is likely due to the lack of armour in the UmbSim model, as well as some contribution from UmbSim's neglecting of proximity effects. The presence armour in the form of steel wires with a high permeability would effectively dampen the parallel resonance peaks.

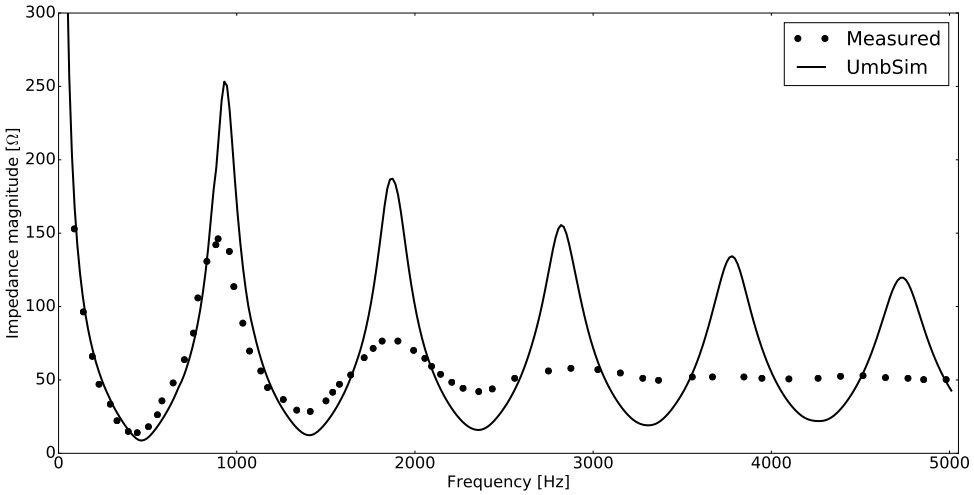
Even though a pipe-type representation of the armour is physically wrong, it is interesting to see if UmbSim is able reproduce the measured impedance plots by varying the parameters and finding a best fit. By letting the surroundings be modelled as an infinite steel pipe with a resistivity equal to that of the steel wires of $2{\cdot}10^{-7}$ Ωm and varying the relative permeability μ_r, the input impedance plots for phase 3 in Fig. 5.31 is created. A best fit for phase 3 is found for $\mu_r \approx 15$. More values for μ_r than those shown in Fig. 5.31 are simulated, but are for practical purposes not shown.

For all six plots the parallel resonance peaks are damped in comparison with Fig. 5.30c. This is due to the lower resistivity of the armour, which allows for currents to flow in it, and hence dampens the input impedance maxima.

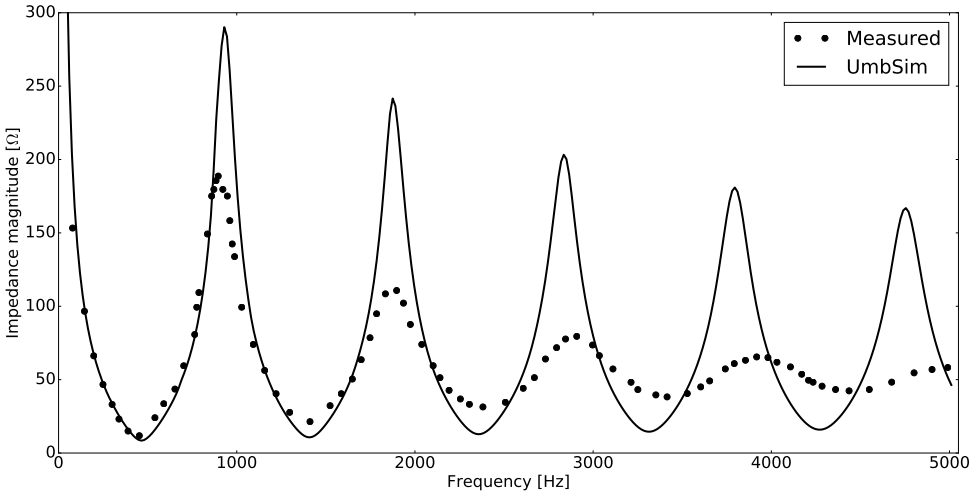
The input impedance of all three power phases with the best-fit armour permeability is plotted in Fig. 5.32. All three plots show an improvement in comparison to Fig. 5.30. The UmbSim simulation is in good agreement with measurements on phase 1 and 2 for frequencies up to around 1500 Hz. After this, descrepancies are evident. It seems that the combination of power phases adjacent to steel duplex tubes and armour leads to a high damping and a decrease in inductance, that UmbSim does not predict. The latter phenomena, and possibly the first, is explained by proximity effects, which UmbSim does not account for.



(a) Phase 1.

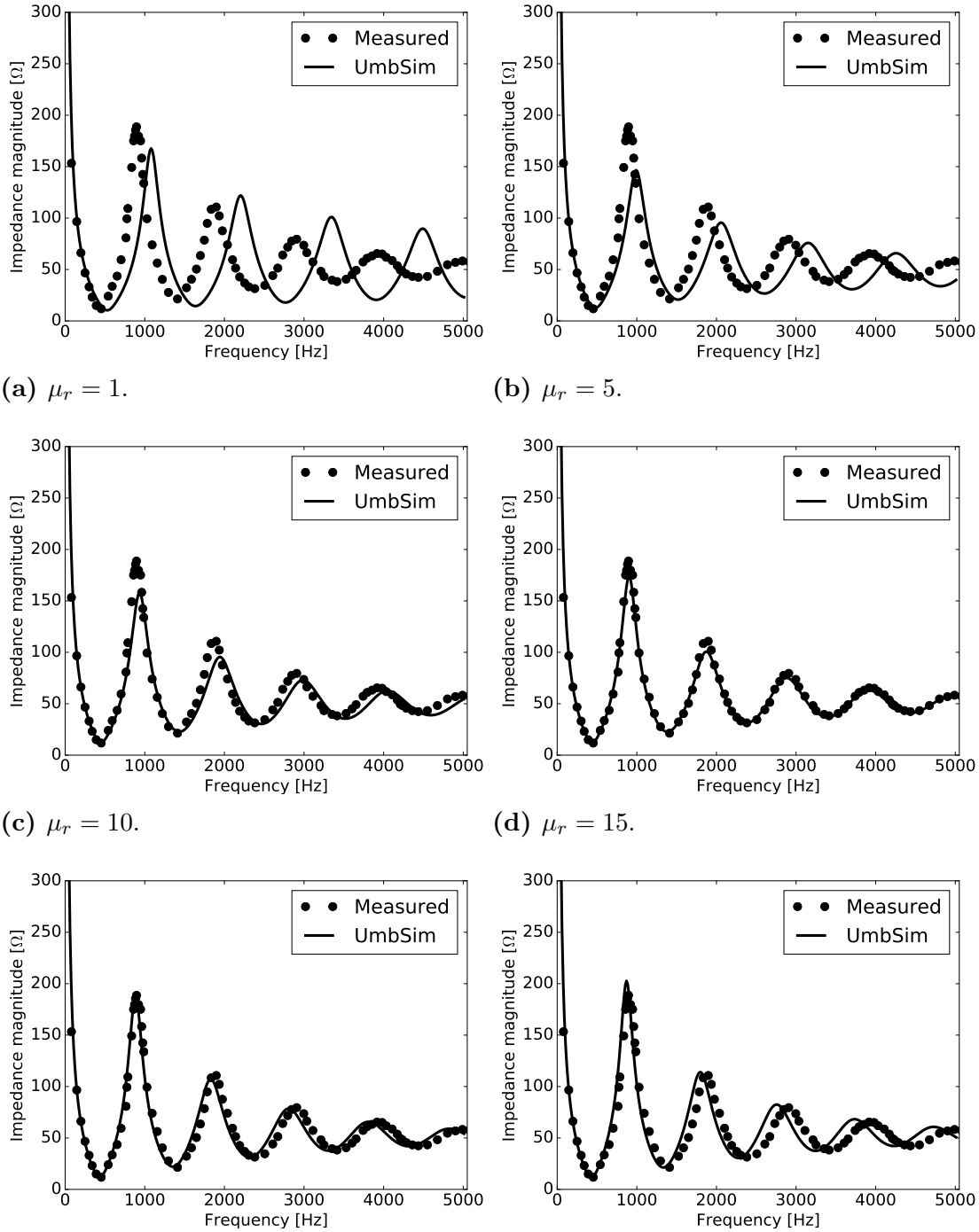


(b) Phase 2.



(c) Phase 3.

Figure 5.30: Umbilical B1 input impedance magnitude spectra for the three power phases. Comparison between UmbSim and measured values provided by Nexans.

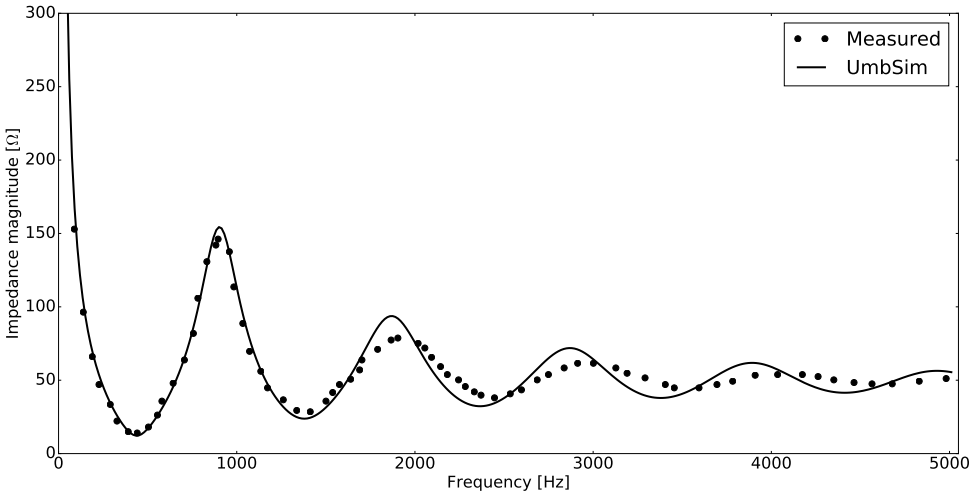


(a) $\mu_r = 1$. **(b)** $\mu_r = 5$.

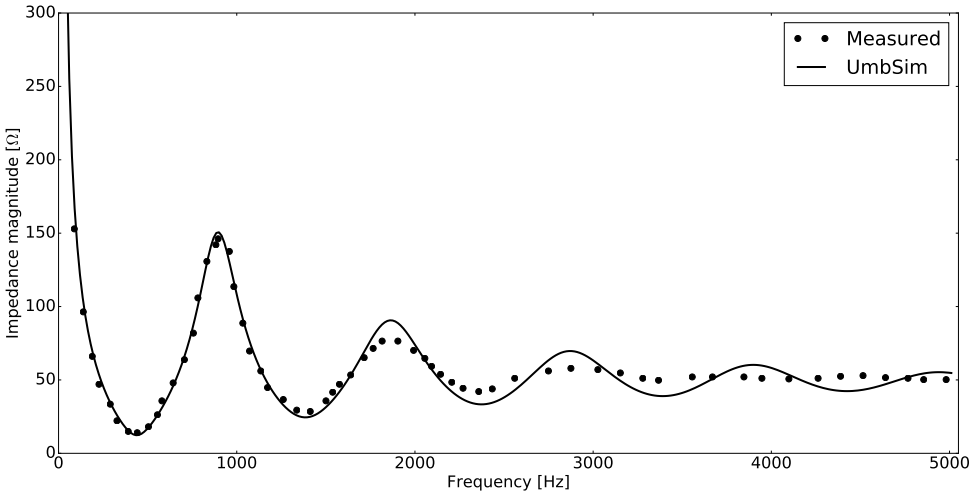
(c) $\mu_r = 10$. **(d)** $\mu_r = 15$.

(e) $\mu_r = 20$. **(f)** $\mu_r = 25$.

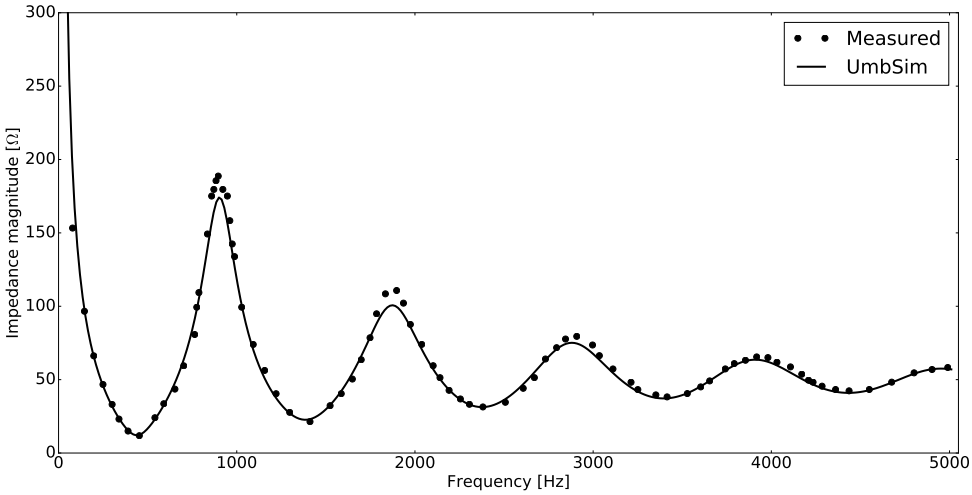
Figure 5.31: UMBSIM simulations of input impedance magnitude spectra for phase 3 in Umbilical B1 with varying values for armour relative permeability μ_r. Armour resistivity used for all plots is $2 \cdot 10^{-7}$ Ωm. Comparison with measured values provided by Nexans.



(a) Phase 1.



(b) Phase 2.



(c) Phase 3.

Figure 5.32: UmbSim simulation of Umbilical B1 input impedance magnitude spectra for the three power phases with $\mu_r = 15$ and $\rho = 2 \cdot 10^{-7}$ Ωm for the armour. Comparison with measured values provided by Nexans.

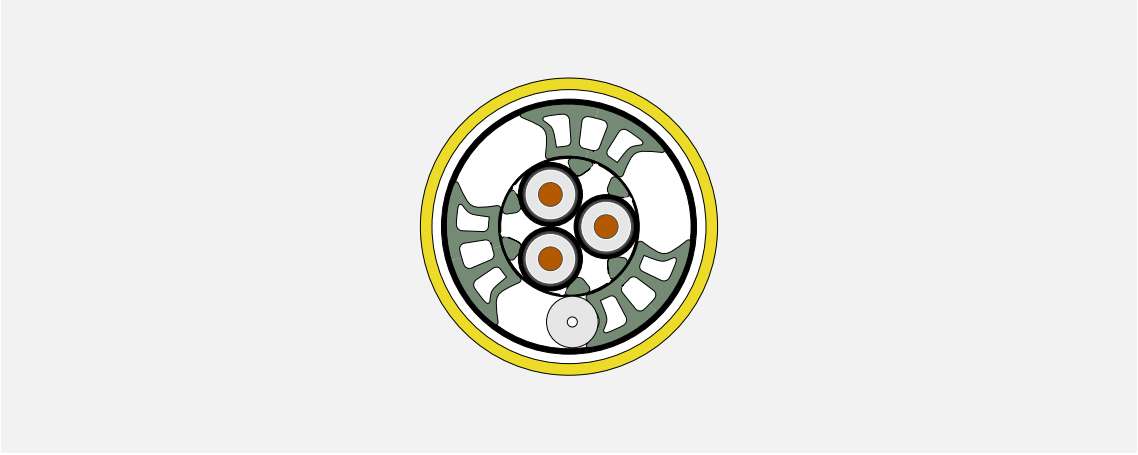


Figure 5.33: Umbilical B2.

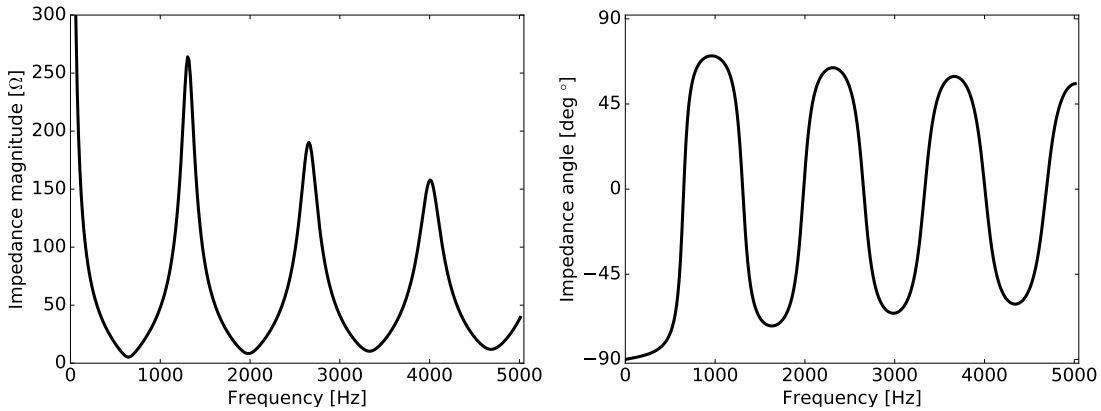
5.2.2 Example B2

The last worked example will be modelling the inner circuit of Umbilical B. This power umbilical, referred to as Umbilical B2, is depicted in Fig. 5.33. First, both the armour and the outer circuit is neglected, and the power umbilical is placed in air with properties as given in Table 5.15. Then modelling a best fit situation as in Example B1 will be done.

Contrary to Umbilical B1, Umbilical B2 has a cyclic symmetric design, and so it can be represented fully by studying only one of the three phases. The terminations for these power phases are given in 5.14.

UmbSim simulation

By running the script in Appendix C.4, per phase input impedance spectra are generated for the inner power circuit. Plots of magnitude and angle are given in Figs. 5.29a and b.



(a) Magnitude. **(b)** Angle.

Figure 5.34: UmbSim simulation of input impedance spectrum for Umbilical B2.

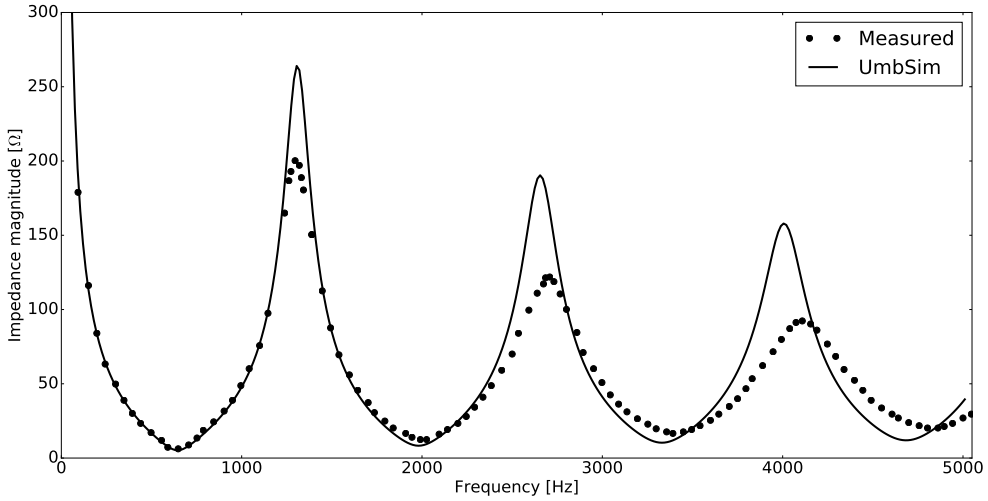
Comparison with measurements

Measurements conducted on the inner power circuit are provided by Nexans as well. Both measured input impedance magnitude and angle are plotted together with the simulation results from UmbSim. The plots can be seen in Fig. 5.35.

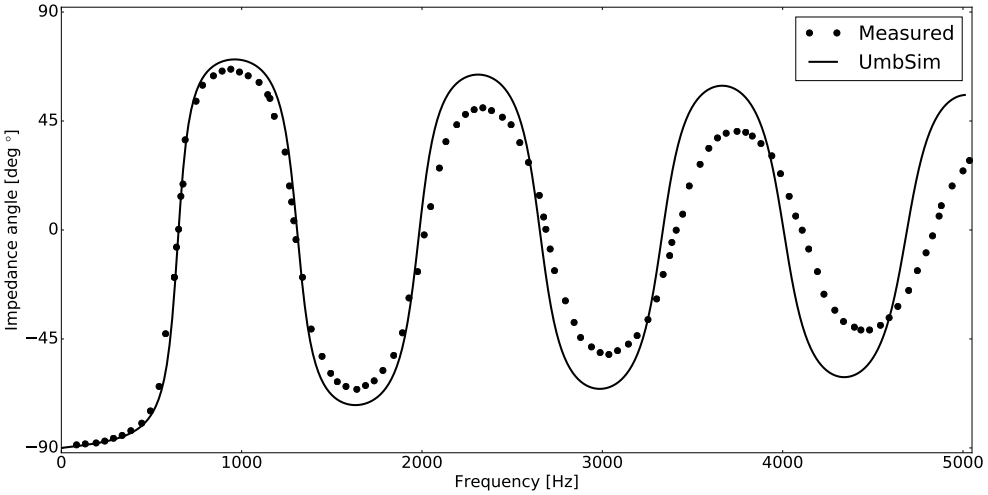
The UmbSim simulations are in a much better agreement with the measured values for the inner circuit than for the outer, with the magnitude spectrum having a relatively small discrepancy for frequencies up to ∼2300 Hz. The better agreement of the simulation of the inner circuit compared to the outer is to be expected, as the armour is further from the inner power circuit and thus will have a lesser influence on the response of the power phases.

However, the neglecting of both armour and proximity effects are prominent in the UmbSim results. By representing the surroundings of Umbilical B2 by an infinite steel pipe with a resistivity of $2 \cdot 10^{-7}$ Ωm and using the best fit relative permeability found in the previous example of 15, the plots in Fig. 5.36 are acquired. Again, the best fit permeability yields an improvement of the simulated spectrum.

It is also likely that the neglecting of the fiber optic elements has a substantial effect on the spectra. The modelling of these would yield a decrease in inductance of the power phases. This would appear in the input impedance spectra for Umbilical B2 as more heavily damped parallel resonance peaks, and a right horizontal shift of the line plots, which would probably lead to a better agreement between UmbSim and measured values.

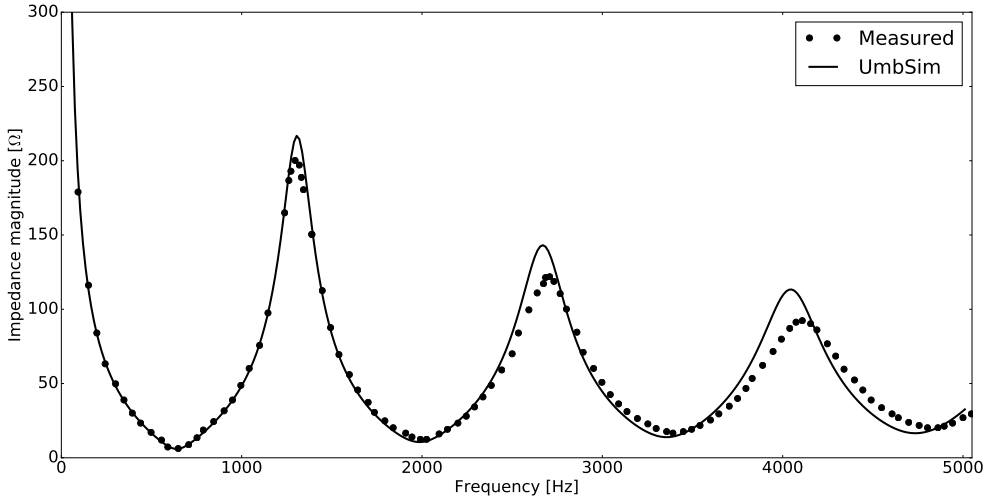


(a) Magnitude.

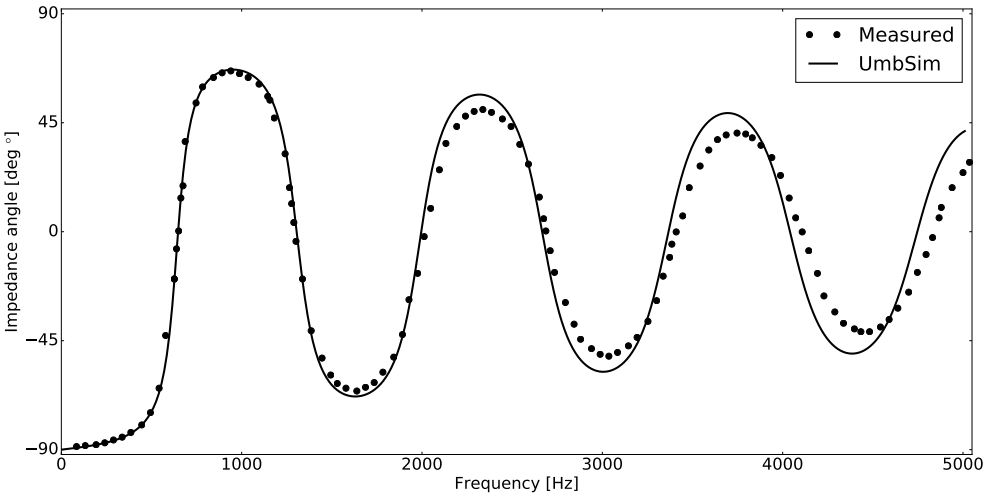


(b) Angle.

Figure 5.35: Umbilical B2 input impedance spectrum. Comparison between UmbSim and measured values provided by Nexans.



(a) Magnitude.



(b) Angle.

Figure 5.36: Umbilical B2 input impedance spectrum with $\mu_r = 15$ and $\rho = 2 \cdot 10^{-7}$ Ωm used for the infinite surrounding pipe. Comparison with measured values provided by Nexans.

6. Concluding remarks and future work

The proposal for this thesis was put forward by Nexans, which wanted a computer program able to predict voltages and currents arising on elements not excited by a voltage source, and to allow a user to conduct harmonic analyses on power umbilical systems.

In this thesis, an electromagnetic model and the computer implementation of it, called UmbSim, has been proposed and validated. The underlying model can be viewed as a combination of two main constituents

1. The analytical solution to the multiconductor telegrapher's equations
2. A general formulation of impedance- and admittance matrices

These two constituents are well-known and utilized in cable modelling, both separately and in combination, for example in Electromagnetic Transients Programs (EMTPs) (Dommel 1986; Martinez-Velasco 2010). The formulation of the parameter matrices is not necessarily that of Ametani (1980), which is the case for EMTP-ATP (Prikler and Høidalen 2009).

The application of the combination of 1. and 2. specifically to power umbilical systems seems novel, but as discussed in Chapter 3, a power umbilical is merely a collection of variations of the general element in Fig. 3.1. The novelty is that it is quite unusual for power cables (for signal cables it is more common) to have the high number and variety of elements in a single enclosure, as power umbilicals do.

6.1 The electromagnetic model and simulations

For the majority of differential equations encountered in mathematical modelling, there are no analytical solutions available. There is, however, an analytical solution to the telegrapher's equations. The choice to use the analytical solution, instead of applying numerical methods for solving the differential equations, seems natural. Instead of solving the differential equations by, i.e. finite-element methods, the problem reduces to solving a set of algebraic equations instead. Generally, this will lead to shorter computation time.

The general formulation for impedance and admittance matrices of Ametani (1980) is presented in literature on cable modelling, for example in Martinez-Velasco

(2010) and F. F. d. Silva and Bak (2013), and parts of the formulation is also commonly used for validating other methods used in cable modelling (Gustavsen et al. 2009). It does, however, have some weaknesses, as seen in the previous chapter. The two prominent observed weaknesses are related to

- Proximity effects
- Modelling of armour comprised of steel wires

There could be some more subtle weaknesses due to the simplifications listed in the beginning of Chapter 3, but these are of less concern, as the two major issues are in focus. For these two prominent weaknesses, some observations were made from the results presented in Chapter 5.

The first is that the formulation of impedance and admittance matrices of Ametani (1980) is not proximity-aware. The deviation due to the lack of modelling of proximity effects is observed in Fig. 5.18, where UmbSim were compared to Flux2D. The discrepancies are prominent for higher frequencies. Based on this observation, proximity effects should not be neglected.

The second prominent weakness observed in the example problems is a lack in capability to model armour comprised of steel wires. The substantial influence of the armour is observed in Example B1 and B2 in the previous chapter, and shows that armour clearly can not be neglected. Since fiber optic elements are neglected, this is also thought to contribute to the observed discrepancies.

By varying the values of the relative permeability of the surroundings and setting the resistivity of the surroundings equal to that of steel, a best fit approximation in terms of input impedance spectra was found for phase 3 in Umbilical B1, see Fig. 5.32. This best fit approximation was used to simulate the input impedance spectra for the other phases of Umbilical B1, as well as for the power phases in Umbilical B2. The best fit simulations were in better agreement with measurements than the unarmoured simulation, but substantial discrepancies are still observed.

6.1.1 Further development of the model

There are some weaknesses in the electromagnetic model that should be addressed

- Proximity effects
- Modelling of armour wires
- Modelling fiber optic elements and and electrical quads

Proximity effects

There are several analytic approaches that could make the proposed model for power umbilicals proximity aware (Kane 1994; Kane, Ahmad, and Auriol 1995; D. d. Silva, Femaindez, and Rivas 2006). They could possibly be relatively simple to include in UmbSim, but a more detailed study must be conducted before conclusions can be drawn.

Modelling of armour wires

Even though the best fit permeability applied to Examples B1 and B2 are found to yield relatively good results for one specific case, it would be interesting to see if the infinite pipe-thickness representation (or the finite pipe-thickness representation) could be tweaked to model a stranded armour adequately well. This could for example be done by estimating the effective permeability of the stranded armour by applying magnetic circuit theory and finding the perturbed resistivity of the stranded armour if it were to be represented as a pipe.

A physically more correct solution would be to use an analytic formulation of the impedance of these armour wires, as proposed by Hatlo et al. (2015). In this formulation, the twisting of the armour wires is taken into account. Numerically this would perhaps be a more difficult task, due to the large number of armour wires.

Modelling fiber optic elements and and electrical quads

The model should also include formulations for armour of fiber optic elements and electrical quad cables.

The armour of the fiber optic elements could be modelled similar to steel duplex tubes, but with no insulating plastic sheath, as they are in a semi-conductive material which provides a good path for transverse currents.

The electrical quads on the other hand can be viewed as a collection of four core conductors, which can be derived directly from a set of the general elements presented in Chapter 3. For electrical quads there are electric fields between their conductors, and hence the capacitance matrix will not be diagonal.

Other topics

A simple algorithm to model a known harmonic voltage spectrum has been applied to Example A3, which can be tweaked to instead model a known harmonic current spectrum, e.g. if one wants to model a non-linear load such as diode bridge rectifiers or other power electronic converters.

6.2 The computer program package

The program package that has been developed, called UmbSim, has worked well when modelling the example problems in the previous chapter; It is easy to use and has an intuitive structure. However, it is still considered to be in developmental stage, even though the most important parts are established.

Note that although not all of the functionality requested by Nexans (listed in the beginning of Chapter 4) have been implemented, most of them are simple to include, such as finding resonance frequencies, or finding maxima of voltages and currents. What areas of UmbSim to focus on has been discussed with Nexans in parallel with development, and therefore this functionality has not been implemented as of yet.

Even though tests are not written for the program package, most of the functionality of UmbSim have been tested through the example problems.

The model only includes formulations for power phases and steel duplex tubes, but the source code has been written based on the most general formulation of the impedance- and admittance matrices, to accommodate further development.

6.2.1 Further development of the program package

The main functionality and core purpose of UmbSim is established, but the program package is still in early development. There are some clear weaknesses present, and to accommodate these some areas of focus for further development are proposed.

First of all, based on the code profiling in Example A2, it is advised that some optimization is applied to the `zpm()` and `zpinner()` methods in the `system.py` module. Some proposed solutions are

- Cythonizing `zpm()` and `zpinner()`.
- Methods that dynamically sets the number of terms in the partial sums in `zpm()` and `zpinner`
- `zpinner()` is a function of the eccentric distance. If other elements have the same eccentric distance, `zpinner()` does not need to be calculated for each element.
- Implementing algorithms for checking if a power umbilical has a cyclic symmetric structure. Then, `zpm()` would not have to be calculated for each mutual coupling.

Secondly, a user of UmbSim is required to have some knowledge of Python. It is therefore proposed that a Graphic User Interface for UmbSim could be created, e.g. in Tkinter or some other package available in Python.

Thirdly, the `simulation.py` module is very simple and provides only a limited number of different plots and outputs. Therefore, more functionality should be added to `simulation.py`. Before this is done, a user can choose to extract the data from simulations as NumPy arrays, and write separate code for whichever analysis one might need to conduct.

Some reported occurrences show the importance of conducting harmonic analysis of power umbilical systems, and therefore built-in functionality for this could be included, where different scenarios could be simulated.

Also, in this thesis, the focus of analysis has been on analysis in frequency domain and steady-state. In the future, UmbSim could be expanded to allow for analysis in time domain and for transient analysis.

References

Schelkunoff, S.A. (1934). *The Electromagnetic Theory of Coaxial Transmission Lines and Cylindrical Shields*. Vol. 13. 4, pp. 532–579.

Brown, G.W. and R.G. Rocamora (1976). "Surge propagation in three-phase pipe-type cables, Part I - Unsaturated pipe". In: *IEEE Transactions of Power Apparatus and Systems* 95.1, pp. 89–95.

Ametani, A. (1980). "A General Formulation of Impedance and Admittance of Cables". In: *IEEE Transactions on Power Apparatus and Systems* PAS-99.3, pp. 902–910. ISSN: 0018-9510. DOI: 10.1109/TPAS.1980.319718.

Dommel, H.W. (1986). *Electromagnetic Transients Program, Reference Manual (EMTP Theory Book*. Bonneville Power Administration.

Kane, M. (1994). "Modèles analytiques originaux pour la détermination des paramètres linéiques des lignes et cables multifilaires parcourus par des signaux large bande". Rapport de Thèse. Ecole Centrale de Lyon.

Ramo, Simon, John R. Whinnery, and Theodore Van Duzer (1994). *Fields and Waves in Communication Electronics, Third Edition*. John Wiley & Sons, Inc. ISBN: 978-0-471-58551-0.

Kane, M., A. Ahmad, and P. Auriol (1995). "Multiwire Shielded Cable Parameter Computation". In: *IEEE Transactions on Magnetics* 31.3, pp. 1646–1649.

Norsk Elektroteknisk Komite (2002). "NEK IEC 61000-2-4, Electromagnetic Compatibility (EMC) - Part 2-4: Environment Compatibility levels in industrial plants for low-frequency conducted disturbances, 2.0 ed". In:

Mohan, Ned, Tore M. Undeland, and William P. Robbins (2003). *Power Electronics - Converters, Applications and Design, Third Edition*. John Wiley & Sons, Inc. ISBN: 978-0-471-22693-2.

Silva, D. da, G. Femaindez, and R. A. Rivas (2006). "Calculation of Frequency-Dependent Parameters of Pipe-Type Cables: Comparison of Methods". In: *2006 IEEE/PES Transmission Distribution Conference and Exposition: Latin America*, pp. 1–6. DOI: 10.1109/TDCLA.2006.311519.

Paul, Clayton R. (2008). *Analysis of Multiconductor Transmission Lines, Second Edition*. International series of monographs on physics. John Wiley & Sons, Inc. ISBN: 978-0-470-13154-1.

Gustavsen, B. et al. (2009). "A Finite-Element Approach for Calculating Electrical Parameters of Umbilical Cables". In: *IEEE Transactions on Power Delivery* 24.4, pp. 2375–2384. ISSN: 0885-8977. DOI: 10.1109/TPWRD.2009.2028481.

Prikler, László and Hans Kristian Høidalen (2009). *ATPDRAW version 5.6 Users' Manual*.

Martinez-Velasco, Juan A. (2010). *Power System Transients Parameter Determination*. CRC Press. ISBN: 978-1-4200-6529-9.

Patel, U. R., B. Gustavsen, and P. Triverio (2013a). "An Equivalent Surface Current Approach for the Computation of the Series Impedance of Power Cables with Inclusion of Skin and Proximity Effects". In: *IEEE Transactions on Power Delivery* 28.4, pp. 2474–2482. ISSN: 0885-8977. DOI: 10.1109/TPWRD.2013.2267098.

— (2013b). "MoM-SO: A fast and fully-automated method for resistance and inductance computation in high-speed cables". In: *2013 17th IEEE Workshop on Signal and Power Integrity*, pp. 1–4. DOI: 10.1109/SaPIW.2013.6558323.

Silva, Filipe Faria da and Claus Leth Bak (2013). *Electromagnetic Transients in Power Cables*. Power Systems. Springer. ISBN: 978-1-4471-5235-4.

Garvik, Øyvind (2015). "Analysis of Harmonic Conditions in Subsea Electrical Power Systems for Oil and Gas Installations". MA thesis. Norwegian University of Science and Technology.

Hatlo, Marius et al. (2015). "Accurate analytical formula for calculation of sheath and armour losses of three core submarine cables". In: Versailles.

Patel, U. R. and P. Triverio (2016). "Accurate Impedance Calculation for Underground and Submarine Power Cables Using MoM-SO and a Multilayer Ground Model". In: *IEEE Transactions on Power Delivery* 31.3, pp. 1233–1241. ISSN: 0885-8977. DOI: 10.1109/TPWRD.2015.2469599.

A. Surface impedance of a solid conductor

In this appendix, the formula for the surface impedance of a solid, cylindrical conductor will be derived. Consider Maxwell's curl equations

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{A.1}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \mathbf{J}_D \tag{A.2}$$

where $\mathbf{E}$ is the electric field intensity, $\mathbf{B}$ is the magnetic flux density, $\mathbf{H}$ is the magnetic field intensity, $\mathbf{J}$ is the conduction current and $\mathbf{J}_D$ is the displacement current.

Let a solid, cylindrical conductor placed concentrically to the z-axis carry a current I. For a good conductor, displacement currents are negligible, so the above equations become

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{A.3}$$

$$\nabla \times \mathbf{B} = \mu\sigma\mathbf{E} \tag{A.4}$$

where the relations $\mathbf{H} = \mu\mathbf{B}$ and $\mathbf{J} = \sigma\mathbf{E}$ are used, with μ and σ being the permeability and the conductivity of the conductor, respectively. Let the current be time-harmonic, so that Maxwell's equation take on the time-harmonic form

$$\nabla \times \mathbf{E}_s = -j\omega\mathbf{B}_s \tag{A.5}$$

$$\nabla \times \mathbf{B}_s = \mu\sigma\mathbf{E}_s \tag{A.6}$$

where subscript s denote that the components of the vector fields are phasor quantities.

Further noting that there is only a longitudinal electric field inside the wire, and only a transverse, azimuthal magnetic field, so that, in cylindrical coordinates

$$\mathbf{E}_s = E_z(\rho)\mathbf{a}_z \tag{A.7}$$

$$\mathbf{B}_s = B_\phi(\rho)\mathbf{a}_\phi \tag{A.8}$$

where $\mathbf{a}_z$ and $\mathbf{a}_\phi$ are the unit vector in the z and azimuthal ϕ direction, respectively. Then the only remaining components of Eqs. (A.5) and (A.6) are

$$\frac{dE_z(\rho)}{d\rho} = j\omega B_\phi(\rho) \tag{A.9}$$

$$\frac{1}{\rho}\frac{d(\rho B_\phi(\rho))}{d\rho} = \mu\sigma E_z(\rho) \tag{A.10}$$

which are coupled. By substituting Eq. (A.9) into (A.10) to eliminate $B_\phi(\rho)$, one gets the ordinary differential equation

$$\frac{1}{\rho}\frac{d}{d\rho}\Big[\rho\frac{d(E_z(\rho))}{d\rho}\Big] - m^2 E_z(\rho) = 0 \tag{A.11}$$

where

$$m = \sqrt{j\omega\mu\sigma} \tag{A.12}$$

is the reciprocal of the *complex penetration depth* of the conductor. Equation (A.11) is known as the modified Bessel's equation and has the general solution

$$E_z(\rho) = AI_0(m\rho) + BK_0(m\rho) \tag{A.13}$$

where A and B are undetermined coefficients, I_0 is the modified Bessel function of the first kind and zeroth order and K_0 is the modified Bessel function of the second kind and zeroth order. The second of these goes to infinity when $\rho \to 0$, so $B = 0$ is chosen and the electric field is given by

$$E_z(\rho) = AI_0(m\rho) \tag{A.14}$$

The accompanying magnetic field inside the conductor is given by substituting the solution above into Eq. (A.9) to get

$$B_\phi(\rho) = \frac{mA}{j\omega} I_1(m\rho) \tag{A.15}$$

where I_1 is the modified Bessel function of the first kind and first order. Note that to get this result, the relation $I_0'(x) = I_1(x)$ has been used along with the chain rule.

The *surface impedance* of the conductor with radius r_1 is now defined to be the surface value of the electric field divided by the total current flowing in the conductor, that is

$$\mathrm{z}_{c\ outer}(\omega) = \frac{E_z(r_1)}{I} \tag{A.16}$$

where I is found from the integral form of Ampere's law as

$$\mu I = \mu I_{enc} = r_1 B_\phi(r_1) \oint d\phi = 2\pi r_1 B_\phi(r_1) \tag{A.17}$$

Hence, the surface impedance of a circular conductor is given by

$$\mathrm{z}_{c\ outer} = \frac{j\omega\mu}{2\pi r_1 m}\frac{I_0(mr_1)}{I_1(mr_1)} \tag{A.18}$$

or

$$\mathrm{z}_{c\ outer} = \frac{\rho_c m}{2\pi r_1}\frac{I_0(mr_1)}{I_1(mr_1)} \tag{A.19}$$

where ρ_c is the resistivity of the conductor and is equal to the reciprocal of σ.

B. UmbSim

B.1 system.py

```
1  # -*- coding: utf-8 -*-
2
3  """
4  This module in UmbSim handles the system consisting of a source, a power
5  umbilical and a load. The main task of system.py is to calculate the impedance
6  and admittance matrix at a frequency f. A user interacts with this module
7  by
8
9  * Defining the geometric and electromagnetic properties of elements and
10   the surroundings through the set_parameters() methods
11 * Creating instances of source, power umbilical and load
12 * Retrieving Z and Y matrices if needed
13
14 et cetera.
15 """
16
17 __author__ = 'Martin Hovde'
18 __email__ = 'martin.hovde@nmbu.no or hovde.martin@gmail.com'
19
20 import math
21 import cmath
22 import numpy as np
23 import scipy.constants as sc
24 import scipy.special as ss
25
26
27 class PredeterminedParametersUmbilical(object):
28     """
29     If parameter matrices are predetermined, use this class instead of
30     PowerUmbilical to create an instance of the power umbilical.
31     """
32
33     def __init__(self, length=None, L=None, C=None, R=None):
34         """
35         :param length: Length of power umbilical
36         :param L: Predetermined inductance matrix
37         :param C: Predetermined capacitance matrix
38         :param R: Predetermined resistance matrix
39         """
40         self.length = length if length else None
41         self.L = L if L else None
42         self.C = C if C else None
43         self.R = R if R else None
44
45     def impedance_matrix(self, f):
46         """
47         Calculates impedance matrix at frequency f.
48         :param f: Frequency
49         :return: Per unit length impedance matrix
50         """
51         return self.R + 1j * 2 * sc.pi * f*self.L
```

```
52
53      def admittance_matrix(self, f):
54          """
55          Calculates admittance matrix at frequency f.
56          :param f: Frequency
57          :return: Per unit length admittance matrix
58          """
59          return 1j * 2 * sc.pi * f * self.C
60
61
62  class Umbilical(object):
63      """
64      Class of power umbilical. Creates instances of all elements and stores them
65      in a matrix.
66      """
67
68      def __init__(self, elements, length):
69          """
70          :param elements: Nested list of elements in power umbilical given as
71
72                           ['power phase', position] or
73                           ['steel tube', position]
74
75                           with position = (d, rho) [metres, degrees]
76          :param length: Length of power umbilical
77          """
78          self.elements = elements
79          self.n = self.matrices_size
80          self.length = length
81          self.instance_matrix = self.element_instance_matrix
82          self.C = self.capacitance_matrix
83
84      def impedance_matrix(self, f):
85          """
86          Calculates impedance matrix at frequency f.
87          :param f: Frequency
88          :return: Per unit length impedance matrix
89          """
90          z_mat = np.zeros(shape=(self.n, self.n), dtype=np.complex_)
91          p, k = 0, 0
92          for i, row in enumerate(self.instance_matrix):
93              for j in range(i+1):
94                  z_sub = self.instance_matrix[i][j].impedance(f)
95                  if z_sub.size > 1:
96                      m, n = z_sub.shape
97                  else:
98                      m, n = 1, 1
99                  z_mat[p:p+m, k:k+n] = z_sub
100                 if i != j:
101                     z_mat[k:k+n, p:p+m] = z_sub.T
102                 k += n
103             k = 0
104             p += m
105         return np.array(z_mat)
106
107     def admittance_matrix(self, f):
108         """
109         Calculates admittance matrix at frequency f.
110         :param f: Frequency
111         :return: Per unit length admittance matrix
112         """
113         return 2j * sc.pi * f * self.C
114
115     @property
116     def matrices_size(self):
117         """
118         Calculates the dimensions of the impedance and admittance matrix.
119         :return: Dimension of impedance and admittance matrix.
120         """
121         n = 0
```

```
122         for elem in self.elements:
123             if elem[0] == 'power phase' and PowerPhase.params['screen']:
124                 n += 2
125             else:
126                 n += 1
127         return n
128 
129     @property
130     def element_instance_matrix(self):
131         """
132         Creates a matrix (nested list) with instances of each element in
133         the power umbilical.
134         :return: Element instance matrix
135         """
136         n = len(self.elements)
137         inst_mat = [[0 for _ in range(n)] for _ in range(n)]
138 
139         for i, elem in enumerate(self.elements):
140             if elem[0] == 'power phase':
141                 inst_mat[i][i] = PowerPhase(elem[1])
142             elif elem[0] == 'steel tube':
143                 inst_mat[i][i] = SteelTube(elem[1])
144             else:
145                 raise ValueError('Element does not exist in library')
146 
147         for i, elem in enumerate(self.elements):
148             for j in range(i+1):
149                 if i != j:
150                     inst_mat[i][j] = inst_mat[j][i] = MutualCoupling(
151                         inst_mat[i][i], inst_mat[j][j])
152         return inst_mat
153 
154     @property
155     def capacitance_matrix(self):
156         """
157         Calculates the capacitance matrix for the power umbilical.
158         :return: Capacitance matrix
159         """
160         p_matrix = np.zeros(shape=(self.n, self.n))
161         k = 0
162         for i, _ in enumerate(self.instance_matrix):
163                 p_sub = self.instance_matrix[i][i].pot_coeff
164                 m = len(p_sub)
165                 if m > 1:
166                     p_matrix[k:k + m, k:k + m] = p_sub
167                 else:
168                     p_matrix[k, k] = p_sub
169                 k += m
170         return np.linalg.inv(p_matrix)
171 
172 
173 class Pipe(object):
174     """
175     Class of surroundings ("pipe").
176     """
177 
178     params = {'mu_r_p': 1,
179               'rho_pipe': 0.3,
180               'd_pipe': 80e-03}
181 
182     def __init__(self, n=25):
183         """
184         :param n: Upper limit of partial sum in Eqs. (3.15) and (3.16)
185         """
186         self.r_pi = self.params['d_pipe'] / 2
187         self.n = n
188 
189     def zpinner(self, f, d):
190         """
191         Given by Eq. (3.15).
```

```
192         :param f: Frequency
193         :param d: Eccentric position of element
194         :return: Impedance of pipe inner surface
195         """
196         mu_r_p = self.params['mu_r_p']
197         m_p = self.m_pipe(f)
198         w = 2 * sc.pi * f
199
200         # Modified Bessel function K
201         k_0rpi = ss.kv(0, m_p * self.r_pi)
202         k_1rpi = ss.kv(1, m_p * self.r_pi)
203
204         z_pi1 = (1 / (m_p * self.r_pi)) * (k_0rpi / k_1rpi)
205
206         z_pi2 = 2 * sum([((d / self.r_pi) ** (2 * k)) * (
207             1. / (k * (1. + mu_r_p) + m_p * self.r_pi *
208                 (ss.kv(k-1, m_p * self.r_pi) / ss.kv(k, m_p * self.r_pi))))
209                                 for k in range(1, self.n+1)])
210         return ((1j * w * sc.mu_0 * mu_r_p) / (2 * sc.pi)) * (z_pi1 + z_pi2)
211
212     @classmethod
213     def set_parameters(cls, parameters):
214         """
215         Allows users to set parameters for surroundings.
216         :param parameters: Dictionary with parameters
217         :return: None
218         """
219         if any(k not in cls.params for k in parameters.iterkeys()):
220             raise KeyError('New parameters not allowed.')
221         if not all(type(v) is int or type(v) is float for v in
222                    parameters.itervalues()):
223             raise ValueError('Parameter values must be numbers.')
224         cls.params.update(parameters)
225
226     def m_pipe(self, f):
227         """
228         Given by Eq. (2.36).
229         :param f: Frequency
230         :return: Reciprocal of pipe's complex penetration depth
231         """
232         return cmath.sqrt((2j * sc.pi * f * sc.mu_0) / self.params['rho_pipe'])
233
234
235 class MutualCoupling(Pipe):
236     """
237     Class of mutual couplings. Describes how two elements interact through
238     mutual impedances with respect to the surroundings inner surface.
239     """
240
241     def __init__(self, elem_i, elem_j):
242         """
243         :param elem_i: Instance of element i
244         :param elem_j: Instance of element j
245         """
246         Pipe.__init__(self)
247         self.elem_i = elem_i
248         self.elem_j = elem_j
249         self.theta_ij = self.elem_i.phi - self.elem_j.phi
250
251     def impedance(self, f):
252         """
253         Calculates the mutual impedance sub-matrices with correct dimensions.
254         :param f: Frequency
255         :return: Mutual impedance sub-matrix as array
256         """
257         zpm = self.zpm(f)
258         impedance = [[zpm for _ in range(self.elem_j.sub_mat_size)]
259                      for _ in range(self.elem_i.sub_mat_size)]
260         return np.array(impedance)
261
```

```
262     def zpm(self, f):
263         """
264         Given by Eq. (3.16).
265         :param f: Frequency
266         :return: Mutual impedance with respect to surroundings inner surface
267         """
268         w = 2 * sc.pi * f
269         d_i = self.elem_i.d
270         d_j = self.elem_j.d
271         m_p = self.m_pipe(f)
272 
273         # Modified Bessel functions I and K
274         k_0rpi = ss.kv(0, m_p * self.r_pi)
275         k_1rpi = ss.kv(1, m_p * self.r_pi)
276 
277         zpm1 = math.log(self.r_pi / math.sqrt(
278             d_i**2 + d_j ** 2 - 2 * d_i * d_j * math.cos(self.theta_ij)))
279 
280         zpm2 = self.params['mu_r_p'] * (1 / (m_p * self.r_pi)) * \
281             (k_0rpi / k_1rpi)
282 
283         zpm3 = sum([((d_i * d_j) / (self.r_pi ** 2)) ** k * math.cos(
284             k * self.theta_ij) * ((2 * self.params['mu_r_p']) / (
285                     k * (1. + self.params['mu_r_p']) + m_p * self.r_pi * (
286                         ss.kv(k - 1, m_p * self.r_pi) /
287                         ss.kv(k, m_p * self.r_pi))) - float(1) / k)
288                     for k in range(1, self.n+1)])
289 
290         zpm = ((1j * w * sc.mu_0) / (2 * sc.pi)) * (zpm1 + zpm2 + zpm3)
291         return zpm
292 
293 
294 class GeneralElement(object):
295     """
296     Abstract class of a general element in a power umbilical.
297     """
298 
299     def __init__(self, position):
300         """
301         :param position: Position as (d, rho) [metres, degrees]
302         """
303         self.d = position[0]
304         self.phi = math.radians(position[1])
305         self.pipe = Pipe()
306 
307     def zcouter(self, r1, f):
308         """
309         Given by Eq. (3.9).
310         :param r1: Core radius
311         :param f: Frequency
312         :return: Impedance of core outer surface
313         """
314         rho = self.conductor_resistivity
315         m_c = self.m(f, rho)
316         i_0 = ss.iv(0, m_c * r1)
317         i_1 = ss.iv(1, m_c * r1)
318         zcouter = ((rho * m_c) / (2 * sc.pi * r1)) * (i_0 / i_1)
319         return zcouter
320 
321     def zcsinsul(self, r1, r2, f):
322         """
323         Given by Eq. (3.10).
324         :param r1: Core radius
325         :param r2: Outer semi-con layer outer radius
326         :param f: Frequency
327         :return: Impedance of core-screen insulation
328         """
329         w = 2 * sc.pi * f
330         zcsinsul = ((1j * w * sc.mu_0) / (2 * sc.pi)) * math.log(r2 / r1)
331         return zcsinsul
```

```
332
333     def zsinner(self, r2, r3, f):
334         """
335         Given by Eq. (3.11).
336         :param r2: Screen inner radius
337         :param r3: Screen outer radius
338         :param f: Frequency
339         :return: Impedance of screen's inner surface
340         """
341         rho = self.params['rho_screen']
342         m_s = self.m(f, rho)
343
344         # Modified Bessel functions In and Kn
345         i_0r2 = ss.iv(0, m_s * r2)
346         i_1r2 = ss.iv(1, m_s * r2)
347         i_1r3 = ss.iv(1, m_s * r3)
348         k_0r2 = ss.kv(0, m_s * r2)
349         k_1r2 = ss.kv(1, m_s * r2)
350         k_1r3 = ss.kv(1, m_s * r3)
351         zsinner = ((rho * m_s) / (2 * sc.pi * r2)) \
352             * ((i_0r2 * k_1r3 + i_1r3 * k_0r2) /
353                (i_1r3 * k_1r2 - i_1r2 * k_1r3))
354         return zsinner
355
356     def zsmutual(self, r2, r3, resistivity, f):
357         """
358         Given by Eq. (3.12).
359         :param r2: Screen- or steel tube inner radius
360         :param r3: Screen- or steel tube outer radius
361         :param resistivity: Resistivity of screen- or steel tube
362         :param f: Frequency
363         :return: Mutual impedance between screen inner and outer surface.
364         """
365         rho = resistivity
366         m_s = self.m(f, rho)
367
368         # Bessel functions
369         i_1r2 = ss.iv(1, m_s * r2)
370         i_1r3 = ss.iv(1, m_s * r3)
371         k_1r2 = ss.kv(1, m_s * r2)
372         k_1r3 = ss.kv(1, m_s * r3)
373         zsmutual = (rho / (2 * sc.pi * r2 * r3)) * (
374             1 / (i_1r3 * k_1r2 - i_1r2 * k_1r3))
375         return zsmutual
376
377     def zsouter(self, r2, r3, resistivity, f):
378         """
379         Given by Eq. (3.13).
380         :param r2: Screen- or steel tube inner radius
381         :param r3: Screen- or steel tube outer radius
382         :param resistivity: Resistivity of either screen or steel tube
383         :param f: Frequency
384         :return: Impedance of screen/steel tube outer surface
385         """
386         m_s = self.m(f, resistivity)
387
388         # Modified Bessel functions In and Kn
389         i_0r3 = ss.iv(0, m_s * r3)
390         i_1r2 = ss.iv(1, m_s * r2)
391         i_1r3 = ss.iv(1, m_s * r3)
392         k_0r3 = ss.kv(0, m_s * r3)
393         k_1r2 = ss.kv(1, m_s * r2)
394         k_1r3 = ss.kv(1, m_s * r3)
395
396         zscouter = ((resistivity * m_s) / (2 * sc.pi * r3)) \
397             * ((i_0r3 * k_1r2 + i_1r2 * k_0r3) /
398                (i_1r3 * k_1r2 - i_1r2 * k_1r3))
399         return zscouter
400
401     def zspinsul(self, r3, r4, f):
```

```
402         """
403         Given by Eq. (3.14).
404         :param r3: Screen- or steel tube outer radius
405         :param r4: Sheath outer radius
406         :param f: Frequency
407         :return: Impedance of insulation between screen/steel tube and '
408         surrounding medium
409         """
410         w = 2 * sc.pi * f
411         r_pi = self.pipe.r_pi
412 
413         zspinsul = ((1j * w * sc.mu_0) / (2 * sc.pi)) * \
414                    (math.log(r4 / r3) + math.log((r_pi / r4) * 1 -
415                                                  (self.d / r_pi) ** 2))
416         return zspinsul
417 
418     def zpinner(self, f):
419         """
420         Given by Eq. (3.15).
421         :param f: Frequency
422         :return: Impedance of surrounding medium inner surface
423         """
424         zpinner = self.pipe.zpinner(f, self.d)
425         return zpinner
426 
427     def zcpinsul(self, r1, f):
428         """
429         Given as the sum of Eq. (3.10) and last term on RHS of Eq. (3.14).
430         :param r1: Core radius
431         :param f: Frequency
432         :return: Impedance between core and inner surface of surrounding medium
433         """
434         w = 2 * sc.pi * f
435         r_pi = self.pipe.r_pi
436         zcpinsul = ((1j * w * sc.mu_0) / (2 * sc.pi)) * \
437             math.log((r_pi / r1) * (1 - (self.d / r_pi) ** 2))
438         return zcpinsul
439 
440     @classmethod
441     def set_parameters(cls, parameters):
442         """
443         Allows users to set parameters for elements in a power umbilical.
444         :param parameters: Dictionary with parameters
445         :return: None
446         """
447         if any(k not in cls.params for k in parameters.iterkeys()):
448             raise KeyError('New parameters not allowed.')
449         cls.params.update(parameters)
450 
451     @staticmethod
452     def m(f, rho):
453         """
454         Given by Eq. (2.36).
455         :param f: Frequency
456         :param rho: Resistivity of material
457         :return: Reciprocal of complex penetration depth
458         """
459         return cmath.sqrt((2j * sc.pi * f * sc.mu_0) / rho)
460 
461 
462 class PowerPhase(GeneralElement):
463     """
464     Class of power phases.
465     """
466 
467     params = {'epsilon_r_ins': 2.4,
468               'mu_r_ins': 1,
469               'screen': False,
470               'core_dc_resistance': 0.001935,
471               'rho_cu': 1.84e-08,
```

```
472             'rho_screen': 8.98e-08,
473             'd_core': 1.15e-02,
474             'd_inner_semi_con': 1.35e-02,
475             'd_core_insulation': 2.99e-02,
476             'd_outer_semi_con': 3.33e-02,
477             'd_screen': 3.5e-02,
478             'd_screen_insulation': 5e-05}
479
480     def __init__(self, position):
481         """
482         :param position: Position as (d, rho) [metres, degrees]
483         """
484         GeneralElement.__init__(self, position)
485         self.sub_mat_size = 2 if self.params['screen'] else 1
486         self.r1 = self.params['d_core'] / 2
487         self.r1marked = self.params['d_inner_semi_con'] / 2
488         self.r2 = self.params['d_outer_semi_con'] / 2
489         self.r2marked = self.params['d_core_insulation'] / 2
490         self.r3 = self.params['d_screen'] / 2
491         self.potential_coefficient = self.pot_coeff
492
493     @property
494     def conductor_resistivity(self):
495         """
496         Calculates the resistivity of the conductor as if it were massive
497         with radius r1.
498         :return: Resistivity
499         """
500         return self.params['core_dc_resistance'] * sc.pi * self.r1 ** 2
501
502     @property
503     def pot_coeff(self):
504         """
505         Given by Eq. (3.26).
506         :return: Potential coefficient sub-matrix for core or core-screen
507         """
508         pc = (1 / (2 * sc.pi * self.params['epsilon_r_ins'] * sc.epsilon_0)) \
509             * math.log(self.r2marked / self.r1marked)
510         return np.array([pc])
511
512     def impedance(self, f):
513         """
514         Calculates the internal impedance sub-matrix for power phase.
515         :param f: Frequency
516         :return: Internal impedance sub-matrix as array
517         """
518         zcouter = self.zcouter(self.r1, f)
519         if self.params['screen']:
520             zcsinsul = self.zcsinsul(self.r1, self.r2, f)
521             zsinner = self.zsinner(self.r2, self.r3, f)
522             zsmutual = self.zsmutual(self.r2, self.r3,
523                                      self.params['rho_screen'], f)
524             zsouter = self.zsouter(self.r2, self.r3, self.params['rho_screen'],
525                                    f)
526             zspinsul = self.zspinsul(self.r3, self.r3, f)
527             zpinner = self.zpinner(f)
528
529             z11 = zcouter + zcsinsul + zsinner + zsouter + zspinsul + zpinner \
530                 - 2 * zsmutual
531             z12 = zsouter + zspinsul + zpinner - zsmutual
532             z22 = zsouter + zspinsul + zpinner
533             return np.array([[z11, z12], [z12, z22]])
534         else:
535             zcpinsul = self.zcpinsul(self.r1, f)
536             zpinner = self.zpinner(f)
537
538             z11 = zcouter + zcpinsul + zpinner
539             return np.array([z11])
540
541
```

```
542 class SteelTube(GeneralElement):
543     """
544     Class of steel duplex tubes.
545     """
546 
547     params = {'steel mu_r': 32,
548               'epsilon_r sheath': 2.3,
549               'rho_tube': 8e-07,
550               'd_inner': 12.7e-03,
551               'd_outer': 15.62e-03,
552               'd_sheath': 1.902e-02}
553 
554     def __init__(self, position):
555         """
556         :param position: Position as (d, rho) [metres, degrees]
557         """
558         GeneralElement.__init__(self, position)
559         self.sub_mat_size = 1
560         self.r2 = self.params['d_inner'] / 2
561         self.r3 = self.params['d_outer'] / 2
562         self.r4 = self.params['d_sheath'] / 2
563         self.potential_coefficient = self.pot_coeff
564 
565     @property
566     def pot_coeff(self):
567         """
568         Given by Eq. (3.27).
569         :return: Potential coefficient sub-matrix for steel duplex tubes.
570         """
571         p = (1 / (2 * sc.pi * sc.epsilon_0 * self.params['epsilon_r sheath'])) \
572             * math.log(self.r4 / self.r3)
573         return np.array([p])
574 
575     def impedance(self, f):
576         """
577         Calculates the internal impedance sub-matrix for steel duplex tubes.
578         :param f: Frequency
579         :return: Internal impedance sub-matrix as array
580         """
581         zst = self.zsouter(self.r2, self.r3, self.params['rho_tube'], f) + \
582             self.zspinsul(self.r3, self.r4, f) + self.zpinner(f)
583         return np.array([zst])
584 
585 
586 class FiberOpticElement(GeneralElement):
587     pass
588 
589 
590 class ElectricalQuad(GeneralElement):
591     pass
592 
593 
594 class Source(object):
595     """
596     Class for the source end of power umbilicals.
597     """
598 
599     def __init__(self, vs, Rs, Ls):
600         """
601         :param vs: Voltage sources in source end
602         :param Rs: Resistance in source end
603         :param Ls: Inductance in source end
604         """
605         self.vs = vs
606         self.Rs = Rs
607         self.Ls = Ls
608 
609     def Zs(self, f):
610         """
611         Calculates the source end impedance matrix.
```

```
612         :param f: Frequency
613         :return: Source end impedance matrix
614         """
615         return self.Rs + 2j * sc.pi * f * self.Ls
616 
617 
618 class Load(object):
619     """
620     Class for the load end of power umbilicals.
621     """
622 
623     def __init__(self, vl, Rl, Ll):
624         """
625         :param vl: Voltage sources in load end
626         :param Rl: Resistance in load end
627         :param Ll: Inductance in load end
628         """
629         self.vl = vl
630         self.Rl = Rl
631         self.Ll = Ll
632 
633     def Zl(self, f):
634         """
635         Calculates the load end impedance matrix.
636         :param f: Frequency
637         :return: Load end impedance matrix
638         """
639         return self.Rl + 2j * sc.pi * f * self.Ll
```

B.2 solver.py

```
# -*- coding: utf-8 -*-

"""
This module calculates the specific solution to the multiconductor telegrapher's
equations. It can calculate the solution at either

* A single frequency and for number of positions along the power umbilical
or
* A single position for a nuber of frequencies

depending on what the user wants to calculate. The user can interact with this
module by using the solution_single_f() or solution_single_z() method to extract
the solution as NumPy arrays.
"""

__author__ = 'Martin Hovde'
__email__ = 'martin.hovde@nmbu.no or hovde.martin@gmail.com'

import numpy as np


def solution_single_f(frequency, z_array, umbilical, source, load, sol_type):
    """
    Calculates the solution to the multiconductor telegrapher's equations along
    a power umbilical for one frequency.

    :param frequency: Frequency
    :param z_array: 1D array with positions along power umbilical
    :param umbilical: Instance of power umbilical
    :param source: Instance of source
    :param load: Instance of load
    :param sol_type: string as 'v', 'i' or 'z'
    :return: nD array with voltage, current or impedance for one frequency
             and all positions in z_array
    """
    params_tuple = solution_params(frequency, umbilical, source, load)
    solution_list = [solution(z, params_tuple, sol_type) for z in z_array]
    return np.array(solution_list)


def solution_single_z(frequency_array, z, umbilical, source, load, sol_type):
    """
    Calculates solutions to the multiconductor telegrapher's equations for
    a power umbilical at one position for frequencies in frequency_array.

    :param frequency_array: 1D array with frequencies
    :param z: A position along umbilical, float or integer
    :param umbilical: Instance of power umbilical
    :param source: Instance of source
    :param load: Instance of load
    :param sol_type: String as 'v', 'i' or 'z'
    :return: nD array with voltage, current or impedance for one positions and
             all frequencies in frequency_array
    """
    num_freqs, num_conductors = len(frequency_array), len(load.vl)
    solution_array = np.zeros((num_freqs, num_conductors), dtype=np.complex_)
    for i, f in enumerate(frequency_array):
        params_tuple = solution_params(f, umbilical, source, load)
        solution_array[i] = solution(z, params_tuple, sol_type)
    return solution_array


def solution(z, params_tuple, sol_type):
    """
    Calculates the solution for a frequency at position z.
    :param z: A position along power umbilical
    :param params_tuple: Tuple with ZC, Ti, lambda_vec, ipos and ineg
```

```
68      :param sol_type: String as 'v', 'i' or 'z'
69      :return: Solution for voltage, current or impedance
70               for a frequency f at position z.
71      """
72      Zc, Ti, lambda_vec, ipos, ineg = params_tuple
73      if sol_type == 'v':
74          voltage = Zc.dot(Ti).dot(np.diag(np.exp(-lambda_vec * z)).dot(ipos) +
75                                   np.diag(np.exp(lambda_vec*z)).dot(ineg))
76          return voltage
77      elif sol_type == 'i':
78          current = Ti.dot(np.diag(np.exp(-lambda_vec * z)).dot(ipos) -
79                       np.diag(np.exp(lambda_vec * z)).dot(ineg))
80          return current
81      elif sol_type == 'z':
82          voltage = (Zc.dot(Ti).dot(np.diag(np.exp(-lambda_vec * z)).dot(ipos) +
83                                   np.diag(np.exp(lambda_vec * z)).dot(ineg)))
84          current = Ti.dot(np.diag(np.exp(-lambda_vec * z)).dot(ipos) -
85                           np.diag(np.exp(lambda_vec * z)).dot(ineg))
86          impedance = voltage / current
87          return impedance
88      else:
89          raise ValueError('Need to specify v or i, either \'i\', \'v\' or \'z\'')
90
91
92  def solution_params(f, umbilical, source, load):
93      """
94      Calculates characteristic impedance matrix, transformation matrix Ti,
95      square-root of diagonal matrix (square-root of eigenvalues) and finds the
96      undetermined coefficients i+ and i-.
97
98      :param f: Frequency
99      :param umbilical: Instance of power umbilical
100     :param source: Instance of source
101     :param load: Instace of load
102     :return: Tuple with Zc, Ti, lambda_vec, ipos, ineg
103     """
104     Z, Y, Zs, Zl = calc_param_matrices(f, umbilical, source, load)
105     vs, vl, length = source.vs, load.vl, umbilical.length
106     lambdasquared, Ti = np.linalg.eig(Y.dot(Z))
107     lambda_vec = np.sqrt(lambdasquared)
108     lambda_mat = np.diag(lambda_vec)
109     Zc = np.linalg.inv(Y).dot(Ti).dot(lambda_mat).dot(np.linalg.inv(Ti))
110
111     exp_neg_l = np.diag(np.exp(-lambda_vec * length))
112     exp_pos_l = np.diag(np.exp(lambda_vec * length))
113
114     BCM = np.vstack((np.hstack(((Zc + Zs).dot(Ti), (Zc - Zs).dot(Ti))),
115                      np.hstack(((Zc - Zl).dot(Ti).dot(exp_neg_l),
116                                 (Zc + Zl).dot(Ti).dot(exp_pos_l)))))
117     v = np.append(vs, vl)
118     i = np.linalg.solve(BCM, v)
119     ipos, ineg = np.split(i, 2)
120     return Zc, Ti, lambda_vec, ipos, ineg
121
122
123 def calc_param_matrices(f, umbilical, source, load):
124     """
125     Calculates the per unit impedance- and admittance matrix for the power
126     umbilical, and the source- and load impedance matrices.
127
128     :param f: Frequency
129     :param umbilical: Instance of power umbilical
130     :param source: Instance of source
131     :param load: Instance of load
132     :return: Tuple with Z, Y, Zs, Zl
133     """
134     Z = umbilical.impedance_matrix(f)
135     Y = umbilical.admittance_matrix(f)
136     Zs = source.Zs(f)
137     Zl = load.Zl(f)
```

```
138         return Z, Y, Zs, Zl
```

B.3 simulation.py

```
# -*- coding: utf-8 -*-

"""
This module is the main module a user interacts with when simulating a specific
system. The user can either request the solution in the form of

* Line plots
* NumPy arrays

for voltage, current or impedance. Either for a single position z and an array
of frequencies, or vice versa.
"""

__author__ = 'Martin Hovde'
__email__ = 'martin.hovde@nmbu.no or hovde.martin@gmail.com'


import solver
import numpy as np
import matplotlib.pyplot as plt


class Simulation(object):
    """
    Class for simulating a system consisting of a power umbilical, source and
    load.
    """

    def __init__(self, umbilical, source, load):
        """
        :param umbilical: Instance of power umbilical
        :param source: Instance of source
        :param load: Instance of load
        """
        self.umbilical = umbilical
        self.source = source
        self.load = load

    def plot_solution(self, f, z, elem, sol_type):
        """
        Method for graphing a line plot for voltage, current or impedance.
        Either:
            frequency f scalar and position z array
            frequency f array and position z scalar
        :param f: Frequency
        :param z: Position
        :param elem: Element to plot solution for. Integer from 0 to number of
                     conductors - 1
        :param sol_type: 'v', 'i' or 'z'
        :return: Line plot
        """

        if np.isscalar(f) and np.isscalar(z):
            raise TypeError('f and z cannot both be scalars.')

        if not np.isscalar(f) and not np.isscalar(z):
            raise TypeError('f and z cannot both be 1D arrays.')

        if np.isscalar(f) and not np.isscalar(z):
            solution = solver.solution_single_f(f, z, self.umbilical,
                                                self.source,
                                                self.load, sol_type)[:, elem]
            plt.plot(z / 10 ** 3, np.absolute(solution), color='k', linewidth=2)
            plt.xlabel('z [km]')
        elif not np.isscalar(f) and np.isscalar(z):
            solution = solver.solution_single_z(f, z, self.umbilical,
                                                self.source,
```

```
68                                                  self.load, sol_type)[:, elem]
69              plt.plot(f , np.absolute(solution), color='k', linewidth=3)
70              plt.xlabel('Frequency [Hz]')
71
72          if sol_type == 'v':
73              plt.ylabel('Voltage [V]')
74          elif sol_type == 'i':
75              plt.ylabel('Current [A]')
76          else:
77              plt.ylabel(r'Impedance [$\Omega$]')
78          plt.show()
79
80      def solution(self, f, z, sol_type):
81          """
82          Finds solution for voltage, current or impedance.
83          Either:
84              frequency f scalar and position z array
85              frequency f array and position z scalar
86          :param f: Frequency
87          :param z: Position
88          :param sol_type: 'v', 'i' or 'z'
89          :return: Voltage, current or impedance as NumPy array
90          """
91          if np.isscalar(f) and not np.isscalar(z):
92              solution = solver.solution_single_f(f, z, self.umbilical,
93                                                  self.source,
94                                                  self.load, sol_type)
95          elif not np.isscalar(f) and np.isscalar(z):
96              solution = solver.solution_single_z(f, z, self.umbilical,
97                                                  self.source,
98                                                  self.load, sol_type)
99          else:
100             raise ValueError('z and f cannot both be scalar or 1D-arrays!')
101         return solution
```

C. Scripts for example problems

C.1 Example A1

UmbSim simulations and comparison with analytical calculations

```
# -*- coding: utf-8 -*-

"""
Example A1
Script for:
* Simulating nominal load
"""

__author__ = 'Martin Hovde'
__email__ = 'martin.hovde@nmbu.no'

import cmath, math
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from UmbSim import system, solver
matplotlib.rcParams.update({'font.size': 18})


system.PowerPhase.set_parameters({'d_core': 11.5e-03,
                                  'd_inner_semi_con': 13.5e-03,
                                  'd_core_insulation': 29.5e-03,
                                  'd_outer_semi_con': 37.9e-03,
                                  'core_dc_resistance': 1.93e-04,
                                  'screen': False})

system.Pipe.set_parameters({'d_pipe': 189e-03,
                            'rho_pipe': 0.3,
                            'mu_r_p': 1})

elements = [['power phase', (22.3e-03, 0)],
            ['power phase', (22.3e-03, 120)],
            ['power phase', (22.3e-03, 240)]]

a = cmath.exp(1j * 2 * math.pi / 3)
V = 36 * 10 ** 3 / math.sqrt(3)
vs = np.array([1, a ** 2, a]).transpose() * V
n = 3
Rs = Ls = np.zeros((n, n))
vl = np.zeros((n, 1))
Rl = np.diag(np.repeat(60, n))
Ll = np.diag(np.repeat(0.3, n))

cable_length = 31e03

umb = system.Umbilical(elements, cable_length)
umb2 = system.Umbilical(elements, 1)
source_inst = system.Source(vs, Rs, Ls)
```

```
load_inst = system.Load(vl, Rl, Ll)

z = 0
f_step = 20
z_array = np.linspace(0, cable_length, 1000)
f_array = np.arange(0.001, 5*10**3 + f_step, f_step)


impedance = np.absolute(solver.solution_single_z(f_array, z, umb, source_inst,
                                                 load_inst, 'z')[:, 0])
angle = np.angle(solver.solution_single_z(f_array, z, umb, source_inst,
                                          load_inst, 'z')[:, 0])

pul_impedance = solver.solution_single_z(f_array, z, umb2, source_inst,
                                         system.Load(vl, Rs, Rs), 'z')[:, 0]
R, X = pul_impedance.real, pul_impedance.imag
L = X / (2 * math.pi * f_array)

def z_in(f, r, l):
    w = 2 * math.pi * f
    c = 1.7080454e-10

    gamma = cmath.sqrt((r + 1j * w * l) * 1j * w * c)
    Z_c = cmath.sqrt((r + 1j * w * l) / (1j * w * c))
    Rl = 60
    Ll = 0.3
    Z_L = Rl + 1j * w * Ll
    z_in = Z_c * ((Z_L + Z_c * cmath.tanh(gamma * cable_length)) /
                  (Z_c + Z_L * cmath.tanh(gamma * cable_length)))
    return abs(z_in)

voltage = solver.solution_single_f(50, z_array, umb, source_inst, load_inst,
                                   'v')[:, 0]
current = solver.solution_single_f(50, z_array, umb, source_inst, load_inst,
                                   'i')[:, 0]

z = voltage[0] / current[0]
r = z.real
l = z.imag / (2 * math.pi * 50)
print 'r:', r, 'l:', l

z_in_array = np.vectorize(z_in)(f_array, R, L)
y1 = plt.plot(f_array, z_in_array, 'ko',
              linewidth=3, label='Analytical', markevery=8, ms=8)
y2 = plt.plot(f_array, impedance, color='k', linewidth=3, label='UmbSim')
plt.xlabel('Frequency [Hz]')
plt.ylabel(r'Impedance magnitude [$\Omega$]')
plt.xlim([0, 5.05e03])
plt.ylim([0, 1000])
plt.legend()
plt.show()

y1 = plt.plot(f_array, impedance, color='k', linewidth=3, label='UmbSim')
plt.xlabel('Frequency [Hz]')
plt.ylabel(r'Impedance magnitude [$\Omega$]')
plt.xlim([0, 5.05e03])
plt.ylim([0, 950])
plt.yticks([0, 200, 400, 600, 800, 950])
plt.show()

y1 = plt.plot(f_array, angle * 180 / math.pi, color='k', linewidth=3)
plt.xlabel('Frequency [Hz]')
plt.ylabel(r'Impedance angle [deg $\degree$]')
plt.xlim([0, 5.05e03])
plt.ylim([-92, 92])
plt.yticks([-90, -45, 0, 45, 90])
plt.show()

y2 = plt.plot(z_array / 10**3, np.absolute(voltage) / 10**3, color='k',
              linewidth=3, label='UmbSim')
```

```
plt.xlabel('z [km]')
plt.ylabel('Voltage magnitude [kV]')
plt.xlim([0, 31])
plt.savefig('A_result_phase_voltage_50_hz.pdf')
plt.show()

y2 = plt.plot(z_array / 10**3, np.absolute(current), color='k', linewidth=3,
              label='UmbSim')
plt.xlabel('z [km]')
plt.ylabel('Current magnitude [A]')
plt.xlim([0, 31])
plt.savefig('A_result_phase_current_50_hz.pdf')
plt.show()
```

```
# -*- coding: utf-8 -*-

"""
Example A1
Script for:
* Simulating open load end
"""

__author__ = 'Martin Hovde'
__email__ = 'martin.hovde@nmbu.no'

import cmath, math
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from UmbSim import system, solver
matplotlib.rcParams.update({'font.size': 18})


system.PowerPhase.set_parameters({'d_core': 11.5e-03,
                                  'd_inner_semi_con': 13.5e-03,
                                  'd_core_insulation': 29.5e-03,
                                  'd_outer_semi_con': 37.9e-03,
                                  'core_dc_resistance': 1.93e-04,
                                  'screen': False})

system.Pipe.set_parameters({'d_pipe': 189e-03,
                            'rho_pipe': 0.3,
                            'mu_r_p': 1})

elements = [['power phase', (22.3e-03, 0)],
            ['power phase', (22.3e-03, 120)],
            ['power phase', (22.3e-03, 240)]]

a = cmath.exp(1j * 2 * math.pi / 3)
V = 36 * 10 ** 3 / math.sqrt(3)


vs = np.array([1, a ** 2, a]).transpose() * V
n = 3 # Number of conductors
Rs = Ls = Ll = np.zeros((n, n))
vl = np.zeros((n, 1))
Rl = np.diag(np.repeat(1e07, n))

cable_length = 31e03

umb = system.Umbilical(elements, cable_length)
umb2 = system.Umbilical(elements, 1)
source_inst = system.Source(vs, Rs, Ls)
load_inst = system.Load(vl, Rl, Ll)

z = 0
f = 50
f_step = 20
z_array = np.linspace(0, cable_length, 1000)
f_array = np.arange(0.001, 5*10**3 + f_step, f_step)
```

```
57
58
59 impedance = np.absolute(solver.solution_single_z(f_array, z, umb, source_inst,
60                                                   load_inst, 'z'))
61 angle = np.angle(solver.solution_single_z(f_array, z, umb, source_inst,
62                                           load_inst, 'z'))
63
64 print sum(impedance[:, 0] - impedance[:, 1]) / len(impedance[:, 0]), \
65     sum(impedance[:, 2] - impedance[:, 1]) / len(impedance[:, 0]), \
66     sum(impedance[:, 2] - impedance[:, 0]) / len(impedance[:, 0])
67
68 voltage = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
69                                                 load_inst, 'v')[:, 0])
70 current = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
71                                                 load_inst, 'i')[:, 0])
72
73 C = umb2.capacitance_matrix[0][0]
74 z_matrix = umb2.impedance_matrix(f)
75 R = (z_matrix[0][0].real - z_matrix[0][1].real)
76 L = (z_matrix[0][0].imag - z_matrix[0][1].imag) / (2 * math.pi * f)
77 print 'C', C, 'L', L, 'R', R
78
79
80 def v(f, r, c, l, z):
81     w = 2 * math.pi * f
82     gammasqrd = (r + 1j * w * l) * 1j * w * c
83     gamma = cmath.sqrt(gammasqrd)
84     vs = 36e03 / math.sqrt(3)
85     length = cable_length
86     alpha = 1 / (1 + cmath.exp(gamma * length * 2))
87     return vs * ((1-alpha) * cmath.exp(-gamma * z) + alpha * cmath.exp(gamma *
88                                                                        z))
89
90 y1 = plt.plot(f_array, impedance[:, 0], color='k', linewidth=3, label='UmbSim')
91 plt.xlabel('Frequency [Hz]')
92 plt.ylabel(r'Impedance magnitude [$\Omega$]')
93 plt.xlim([0, 5.05e03])
94 plt.ylim([0, 950])
95 plt.yticks([0, 200, 400, 600, 800, 950])
96 plt.show()
97
98 y1 = plt.plot(f_array, angle[:, 0] * 180 / math.pi, color='k', linewidth=3,
99               label='UmbSim')
100 plt.xlabel('Frequency [Hz]')
101 plt.ylabel(r'Impedance angle [deg $\degree$]')
102 plt.xlim([0, 5.05e03])
103 plt.ylim([-92, 92])
104 plt.yticks([-90, -45, 0, 45, 90])
105 plt.show()
106
107 # Percentage error
108 fig, ax = plt.subplots()
109 v_analytical = abs(np.vectorize(v)(f, R, C, L, z_array))
110 plt.plot(z_array / 10**3, (v_analytical - voltage) * 100 / v_analytical * 10**5,
111          color='k', linewidth=3, label='Analytical')
112 plt.xlabel('z [km]')
113 plt.ylabel(r'Difference [$10^{-5}$ %]')
114 plt.xlim([0, cable_length / 10**3])
115 plt.show()
116
117 # UmbSim and analytical plotted together
118 fig, ax = plt.subplots()
119 plt.plot(z_array / 10**3, voltage / 10**3, color='k', linewidth=3,
120          label='UmbSim')
121 v_analytical = abs(np.vectorize(v)(f, R, C, L, z_array))
122 plt.plot(z_array / 10**3, v_analytical / 10**3, color='k', linestyle='-', ms=9,
123          marker='o', linewidth=3, markevery=500, label='Analytical')
124 ax.get_yaxis().get_major_formatter().set_useOffset(False)
125 plt.xlabel('z [km]')
126 plt.ylabel('Voltage magnitude [kV]')
```

```
127 plt.xlim([0, cable_length / 10**3])
128 plt.legend(loc=2)
129 plt.show()
130
131 y2 = plt.plot(z_array / 10**3, current, color='k', linewidth=3, label='UmbSim')
132 plt.xlabel('z [km]')
133 plt.ylabel('Current magnitude [A]')
134 plt.xlim([0, cable_length / 10**3])
135 plt.savefig('A_result_phase_current_50_hz.pdf')
136 plt.show()
```

```
 1 # -*- coding: utf-8 -*-
 2
 3 """
 4 Example A1
 5 Script for:
 6 * Simulating shorted load end
 7 """
 8
 9 __author__ = 'Martin Hovde'
10 __email__ = 'martin.hovde@nmbu.no'
11
12 import cmath, math
13 import numpy as np
14 import matplotlib
15 import matplotlib.pyplot as plt
16 from UmbSim import system, solver
17 matplotlib.rcParams.update({'font.size': 18})
18
19 system.PowerPhase.set_parameters({'d_core': 11.5e-03,
20                                   'd_inner_semi_con': 13.5e-03,
21                                   'd_core_insulation': 29.5e-03,
22                                   'd_outer_semi_con': 37.9e-03,
23                                   'core_dc_resistance': 1.93e-04,
24                                   'screen': False})
25
26 system.Pipe.set_parameters({'d_pipe': 189e-03,
27                             'rho_pipe': 0.3,
28                             'mu_r_p': 1})
29
30 elements = [['power phase', (22.3e-03, 0)],
31              ['power phase', (22.3e-03, 120)],
32              ['power phase', (22.3e-03, 240)]]
33
34 a = cmath.exp(1j * 2 * math.pi / 3)
35 V = 36 * 10 ** 3 / math.sqrt(3)
36 n = 3
37 vs = np.array([1, a ** 2, a]).transpose() * V
38 Rs = np.zeros((n, n))
39 Ls = np.zeros((n, n))
40 vl = np.zeros((n, 1))
41 Rl = np.diag(np.zeros(n))
42 Ll = np.diag(np.zeros(n))
43
44 cable_length = 31e03
45
46 umb = system.Umbilical(elements, cable_length)
47 source_inst = system.Source(vs, Rs, Ls)
48 load_inst = system.Load(vl, Rl, Ll)
49
50 z = 0
51 f_step = 20
52 z_array = np.linspace(0, cable_length, 1000)
53 f_array = np.arange(0.001, 5*10**3 + f_step, f_step)
54
55
56 impedance = np.absolute(solver.solution_single_z(f_array, z, umb, source_inst,
57                                                  load_inst, 'z')[:, 0])
58 angle = np.angle(solver.solution_single_z(f_array, z, umb, source_inst,
59                                           load_inst, 'z')[:, 0])
```

```
60
61 voltage = solver.solution_single_f(50, z_array, umb, source_inst, load_inst,
62                                     'v')[:, 0]
63 current = solver.solution_single_f(50, z_array, umb, source_inst, load_inst,
64                                     'i')[:, 0]
65
66 plt.plot(f_array, impedance, color='k', linewidth=3)
67 plt.xlabel('Frequency [Hz]')
68 plt.ylabel(r'Impedance magnitude [$\Omega$]')
69 plt.xlim([0, 5.05e03])
70 plt.ylim([0, 950])
71 plt.yticks([0, 200, 400, 600, 800, 950])
72 plt.show()
73
74 plt.plot(f_array, angle * 180 / math.pi, color='k', linewidth=3)
75 plt.xlabel('Frequency [Hz]')
76 plt.ylabel(r'Impedance angle [deg $\degree$]')
77 plt.xlim([0, 5.05e03])
78 plt.ylim([-92, 92])
79 plt.yticks([-90, -45, 0, 45, 90])
80 plt.show()
81
82 plt.plot(z_array / 10**3, np.absolute(voltage) / 10**3, color='k', linewidth=3)
83 plt.xlabel('z [km]')
84 plt.ylabel('Voltage magnitude [kV]')
85 plt.xlim([0, 31])
86 plt.savefig('A_result_phase_voltage_50_hz.pdf')
87 plt.show()
88
89 y2 = plt.plot(z_array / 10**3, np.absolute(current), color='k', linewidth=3)
90 plt.xlabel('z [km]')
91 plt.ylabel('Current magnitude [A]')
92 plt.xlim([0, 31])
93 plt.savefig('A_result_phase_current_50_hz.pdf')
94 plt.show()
```

Comparison with Flux2D

```
 1 # -*- coding: utf-8 -*-
 2
 3 """
 4 Example A1
 5 Script for:
 6 * Comparison with Flux2D
 7 """
 8
 9 __author__ = 'Martin Hovde'
10 __email__ = 'martin.hovde@nmbu.no'
11
12 import cmath, math
13 import numpy as np
14 import matplotlib
15 import matplotlib.pyplot as plt
16 from UmbSim import system, solver
17
18
19 system.PowerPhase.set_parameters({'d_core': 11.5e-03,
20                                   'd_inner_semi_con': 13.5e-03,
21                                   'd_core_insulation': 29.5e-03,
22                                   'd_outer_semi_con': 37.9e-03,
23                                   'core_dc_resistance': 1.93e-04,
24                                   'screen': False})
25
26 system.Pipe.set_parameters({'d_pipe': 189e-03,
27                             'rho_pipe': 0.3,
28                             'mu_r_p': 1})
29
30 elements = [['power phase', (22.3e-03, 0)],
```

```
31              ['power phase', (22.3e-03, 120)],
32              ['power phase', (22.3e-03, 240)]]
33
34 a = cmath.exp(1j * 2 * math.pi / 3)
35 V = 36 * 10 ** 3 / math.sqrt(3)
36 n = 3
37 vs = np.array([1, a ** 2, a]).transpose() * V
38 Rs = np.zeros((n, n))
39 Ls = np.zeros((n, n))
40 vl = np.zeros((n, 1))
41 Rl = np.diag(np.repeat(60, n))
42 Ll = np.diag(np.repeat(0.3, n))
43
44 cable_length = 31e03
45
46 umb = system.Umbilical(elements, cable_length)
47 source_inst = system.Source(vs, Rs, Ls)
48 load_inst = system.Load(vl, Rl, Ll)
49
50 z = 0
51 f_step = 20
52 f_array = np.arange(1, 5*10**3 + f_step, f_step)
53
54 in_data = np.loadtxt('impedance_data_marius_only_power_phases.txt', skiprows=0)
55 f_array_m, impedance_m = in_data[:, 0], in_data[:, 1]
56
57 impedance = np.absolute(solver.solution_single_z(f_array, z, umb, source_inst,
58                                                   load_inst, 'z')[:, 0])
59
60 matplotlib.rcParams.update({'font.size': 24})
61 y1 = plt.plot(f_array, impedance, color='k', linewidth=3, label='UmbSim')
62 y2 = plt.plot(f_array_m, impedance_m, color='k', marker='o', ms=9, linewidth=0,
63               label='Flux2D')
64 plt.xlabel('Frequency [Hz]')
65 plt.ylabel(r'Impedance magnitude [$\Omega$]')
66 plt.xlim([45, 5.1e03])
67 plt.ylim([0, 1000])
68 plt.legend()
69 plt.show()
```

C.2 Example A2

UmbSim simulations

```
# -*- coding: utf-8 -*-

"""
Example A2
Script for:
* Plotting input impedance spectra
* Plotting voltage and current at 50 Hz for different terminations
"""

__author__ = 'Martin Hovde'
__email__ = 'martin.hovde@nmbu.no'

import cmath, math
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from UmbSim import system, solver
matplotlib.rcParams.update({'font.size': 18})


system.PowerPhase.set_parameters({'d_core': 11.5e-03,
                                  'd_inner_semi_con': 13.5e-03,
                                  'd_core_insulation': 29.5e-03,
                                  'd_outer_semi_con': 37.9e-03,
                                  'core_dc_resistance': 1.93e-04,
                                  'screen': False})

system.SteelTube.set_parameters({'d_inner': 1.27e-02,
                                 'd_outer': 1.562e-02,
                                 'd_sheath': 1.902e-02,
                                 'rho_tube': 8e-07,
                                 'steel mu_r': 32})

system.Pipe.set_parameters({'d_pipe': 189e-03,
                            'rho_pipe': 0.3,
                            'mu_r_p': 1})

elements = [['power phase', (22.3e-03, 0)],
            ['power phase', (22.3e-03, 120)],
            ['power phase', (22.3e-03, 240)],
            ['steel tube', (32.34e-03, 60)],
            ['steel tube', (32.34e-03, 180)],
            ['steel tube', (32.34e-03, 300)]]

a = cmath.exp(1j * 2 * math.pi / 3)
V = 36 * 10 ** 3 / math.sqrt(3)
n = 6  # No. of conductors
vs = np.array([1, a ** 2, a, 0, 0, 0]).transpose() * V
Rs = np.zeros((n, n))
Ls = np.zeros((n, n))
vl = np.zeros((n, 1))
Rl = np.diag([60., 60., 60., 0, 0, 0])
Ll = np.diag([0.3, 0.3, 0.3, 0, 0, 0])

cable_length = 31e03

umb = system.Umbilical(elements, cable_length)
source_inst = system.Source(vs, Rs, Ls)
load_inst = system.Load(vl, Rl, Ll)

z = 0
f = 50
f_step = 25
z_array = np.linspace(0, cable_length, 400)
```

```
f_array = np.arange(1, 5*10**3 + f_step, f_step)

# Nominal load
impedance = np.absolute(solver.solution_single_z(f_array, z, umb, source_inst,
                                                 load_inst, 'z')[:, 0])
plt.plot(f_array, impedance, color='k', linewidth=3)
plt.xlabel('Frequency [Hz]')
plt.ylabel(r'Impedance magnitude [$\Omega$]')
plt.ylim([0, 950])
plt.yticks([0, 200, 400, 600, 800, 950])
plt.xlim([0, 5.1e03])
plt.show()

voltage = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
                                               load_inst, 'v')[:, 0])
fig, ax = plt.subplots()
plt.plot(z_array / 10**3, voltage / 10**3, color='k', linewidth=3)
plt.xlabel('z [km]')
plt.ylabel(r'Voltage magnitude [kV]')
ax.get_yaxis().get_major_formatter().set_useOffset(False)
plt.xlim([0, cable_length / 10**3])
plt.show()

current = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
                                               load_inst, 'i')[:, 0])
fig, ax = plt.subplots()
plt.plot(z_array / 10**3, current, color='k', linewidth=3)
plt.xlabel('z [km]')
plt.ylabel(r'Current magnitude [A]')
ax.get_yaxis().get_major_formatter().set_useOffset(False)
plt.xlim([0, cable_length / 10**3])
plt.show()

# Open load end
Rl = np.diag(np.array([1e7, 1e7, 1e7, 0, 0, 0]))
Ll = np.zeros((n, n))
load_inst = system.Load(vl, Rl, Ll)

impedance = np.absolute(solver.solution_single_z(f_array, z, umb, source_inst,
                                                 load_inst, 'z')[:, 0])
plt.plot(f_array, impedance, color='k', linewidth=3)
plt.xlabel('Frequency [Hz]')
plt.ylim([0, 950])
plt.yticks([0, 200, 400, 600, 800, 950])
plt.ylabel(r'Impedance magnitude [$\Omega$]')
plt.xlim([0, 5.1e03])
plt.show()

voltage = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
                                               load_inst, 'v')[:, 0])
fig, ax = plt.subplots()
plt.plot(z_array / 10**3, voltage / 10**3, color='k', linewidth=3)
plt.xlabel('z [km]')
plt.ylabel(r'Voltage magnitude [kV]')
ax.get_yaxis().get_major_formatter().set_useOffset(False)
plt.xlim([0, cable_length / 10**3])
plt.show()

current = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
                                               load_inst, 'i')[:, 0])
fig, ax = plt.subplots()
plt.plot(z_array / 10**3, current, color='k', linewidth=3)
plt.xlabel('z [km]')
plt.ylabel(r'Current magnitude [A]')
ax.get_yaxis().get_major_formatter().set_useOffset(False)
plt.xlim([0, cable_length / 10**3])
plt.show()

# Shorted load end
Rl = np.zeros((n, n))
```

```
135 Ll = np.zeros((n, n))
136 load_inst = system.Load(vl, Rl, Ll)
137
138 impedance = np.absolute(solver.solution_single_z(f_array, z, umb, source_inst,
139                                                   load_inst, 'z')[:, 0])
140 plt.plot(f_array, impedance, color='k', linewidth=3)
141 plt.xlabel('Frequency [Hz]')
142 plt.ylabel(r'Impedance magnitude [$\Omega$]')
143 plt.xlim([0, 5.1e03])
144 plt.show()
145
146 voltage = np.asbolute(solver.solution_single_f(f, z_array, umb, source_inst,
147                                                 load_inst, 'v')[:, 0])
148 fig, ax = plt.subplots()
149 plt.plot(z_array / 10**3, voltage / 10**3, color='k', linewidth=3)
150 plt.xlabel('z [km]')
151 plt.ylabel(r'Voltage magnitude [kV]')
152 ax.get_yaxis().get_major_formatter().set_useOffset(False)
153 plt.xlim([0, cable_length / 10**3])
154 plt.show()
155
156 current = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
157                                                 load_inst, 'i')[:, 0])
158 fig, ax = plt.subplots()
159 plt.plot(z_array / 10**3, current, color='k', linewidth=3)
160 plt.xlabel('z [km]')
161 plt.ylabel(r'Current magnitude [A]')
162 ax.get_yaxis().get_major_formatter().set_useOffset(False)
163 plt.xlim([0, cable_length / 10**3])
164 plt.show()
```

Comparison with analytical calculations

```
 1 # -*- coding: utf-8 -*-
 2
 3 """
 4 Example A2
 5 Script for:
 6 * Comparing duplex voltage at 50 Hz for analytical and UmbSim
 7 """
 8
 9 __author__ = 'Martin Hovde'
10 __email__ = 'martin.hovde@nmbu.no'
11
12 import cmath, math
13 import numpy as np
14 from UmbSim import system, solver
15 import matplotlib
16 import matplotlib.pyplot as plt
17 matplotlib.rcParams.update({'font.size': 24})
18
19 system.PowerPhase.set_parameters({'d_core': 11.5e-03,
20                                   'd_inner_semi_con': 13.5e-03,
21                                   'd_core_insulation': 29.5e-03,
22                                   'd_outer_semi_con': 37.9e-03,
23                                   'core_dc_resistance': 1.93e-04,
24                                   'screen': False})
25
26 system.SteelTube.set_parameters({'d_inner': 1.27e-02,
27                                  'd_outer': 1.562e-02,
28                                  'd_sheath': 1.902e-02,
29                                  'rho_tube': 8e-07})
30
31 system.Pipe.set_parameters({'d_pipe': 189e-03,
32                             'rho_pipe': 0.3,
33                             'mu_r_p': 1})
34
35 elements = [['power phase', (21.88e-03, 0)],
```

```
36              ['power phase', (21.88e-03, 120)],
37              ['power phase', (21.88e-03, 240)],
38              ['steel tube', (31.32e-03, 60)],
39              ['steel tube', (31.32e-03, 180)],
40              ['steel tube', (31.32e-03, 300)]]
41
42 a = cmath.exp(1j * 2 * math.pi / 3)
43 V = 36 * 10 ** 3 / math.sqrt(3)
44 n = 6  # Number of conductors
45 vs = np.array([1, a ** 2, a, 0, 0, 0]).transpose() * V
46 Rs = np.zeros((n, n))
47 Ls = np.zeros((n, n))
48 vl = np.zeros((n, 1))
49 Rl = np.diag([60., 60., 60., 0, 0, 0])
50 Ll = np.diag([0.3, 0.3, 0.3, 0, 0, 0])
51
52 cable_length = 31e03
53
54 umb = system.Umbilical(elements, cable_length)
55 source_inst = system.Source(vs, Rs, Ls)
56 load_inst = system.Load(vl, Rl, Ll)
57
58 f = 50
59 z_step = 20
60 z_array = np.linspace(0, cable_length, 200)
61
62 voltage = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
63                                                load_inst, 'v')[:, 4])
64
65 z_mat = umb.impedance_matrix(50)
66 c = umb.capacitance_matrix[0][0]
67 l = (z_mat[4][0] - z_mat[4][1]).imag / (2 * math.pi * f)
68
69 w = 2 * math.pi * f
70
71
72 def v_analytical(z):
73     v = abs(0.5 * w ** 2 * c * l * 19.8 * 10**3 * (z**2 - z * cable_length))
74     return v
75
76
77 voltage_analytical = np.vectorize(v_analytical)(z_array)
78 y1 = plt.plot(z_array / 10**3, voltage, color='k', linewidth=3, label='UmbSim')
79 y2 = plt.plot(z_array / 10**3, voltage_analytical, 'ko', ms=9, linewidth=0,
80               markevery=10, label='Analytical')
81 plt.xlabel('z [km]')
82 plt.ylabel('Voltage magnitude [V]')
83 plt.xlim([0, 31])
84 plt.legend()
85 plt.show()
```

Comparison with Flux2D

```
 1 # -*- coding: utf-8 -*-
 2
 3 """
 4 Example A2
 5 Script for:
 6 * Input impedance spectrum comparison with Flux2D
 7 """
 8
 9 __author__ = 'Martin Hovde'
10 __email__ = 'martin.hovde@nmbu.no'
11
12 import cmath, math
13 import numpy as np
14 import matplotlib
15 import matplotlib.pyplot as plt
```

```
from UmbSim import system, solver
matplotlib.rcParams.update({'font.size': 24})

system.PowerPhase.set_parameters({'d_core': 11.5e-03,
                                  'd_inner_semi_con': 13.5e-03,
                                  'd_core_insulation': 29.5e-03,
                                  'd_outer_semi_con': 37.9e-03,
                                  'core_dc_resistance': 1.93e-04,
                                  'screen': False})

system.SteelTube.set_parameters({'d_inner': 1.27e-02,
                                 'd_outer': 1.562e-02,
                                 'd_sheath': 1.902e-02,
                                 'rho_tube': 8e-07})

system.Pipe.set_parameters({'d_pipe': 189e-03,
                            'rho_pipe': 0.3,
                            'mu_r_p': 1})

elements = [['power phase', (22.3e-03, 0)],
            ['power phase', (22.3e-03, 120)],
            ['power phase', (22.3e-03, 240)],
            ['steel tube', (32.34e-03, 60)],
            ['steel tube', (32.34e-03, 180)],
            ['steel tube', (32.34e-03, 300)]]

a = cmath.exp(1j * 2 * math.pi / 3)
V = 36 * 10 ** 3 / math.sqrt(3)
n = 6  # Number of conductors
vs = np.array([[1], [a ** 2], [a], [0], [0], [0]])
Rs = Ls = np.zeros((n, n))
vl = np.zeros((n, 1))
Rl = np.diag([60., 60., 60., 0, 0, 0])
Ll = np.diag([0.3, 0.3, 0.3, 0, 0, 0])

cable_length = 31e03

umb = system.Umbilical(elements, cable_length)
source_inst = system.Source(vs, Rs, Ls)
load_inst = system.Load(vl, Rl, Ll)

z = 0
f_step = 20
f_array = np.arange(1, 5*10**3 + f_step, f_step)

in_data = np.loadtxt('example_B_impedance_data_marius.txt', skiprows=0)
f_array_m, impedance_m = in_data[:, 0], in_data[:, 1]

impedance = np.absolute(solver.solution_single_z(f_array, z, umb, source_inst,
                                                 load_inst, 'z')[:, 0])

y1 = plt.plot(f_array, impedance, color='k', linewidth=2, label='UmbSim')
y3 = plt.plot(f_array_m, impedance_m, 'ko', ms=9, linewidth=0, label='Flux2D')
plt.xlabel('Frequency [Hz]')
plt.ylabel(r'Impedance magnitude [$\Omega$]')
plt.xlim([0, 5.1e03])
plt.legend()
plt.show()
```

```
# -*- coding: utf-8 -*-

"""
Example A2
Script for:
* Comparing per unit length parameters from Flux2D with UmbSim
* Comparing corrected input impedance spectra from Flux2D with UmbSim
"""

__author__ = 'Martin Hovde'
__email__ = 'martin.hovde@nmbu.no'
```

```
12
13 import cmath, math
14 import numpy as np
15 import matplotlib
16 import matplotlib.pyplot as plt
17 from UmbSim import system, solver
18 matplotlib.rcParams.update({'font.size': 24})
19
20 system.PowerPhase.set_parameters({'d_core': 11.5e-03,
21                                   'd_inner_semi_con': 13.5e-03,
22                                   'd_core_insulation': 29.5e-03,
23                                   'd_outer_semi_con': 37.9e-03,
24                                   'core_dc_resistance': 1.93e-04,
25                                   'screen': False})
26
27 system.SteelTube.set_parameters({'d_inner': 1.27e-02,
28                                  'd_outer': 1.562e-02,
29                                  'd_sheath': 1.902e-02,
30                                  'rho_tube': 8e-07})
31
32 system.Pipe.set_parameters({'d_pipe': 189e-03,
33                             'rho_pipe': 0.3,
34                             'mu_r_p': 1})
35
36 a = cmath.exp(1j * 2 * math.pi / 3)
37 V = 36 * 10 ** 3 / math.sqrt(3)
38
39 elements = [['power phase', (22.3e-03, 0)],
40             ['power phase', (22.3e-03, 120)],
41             ['power phase', (22.3e-03, 240)],
42             ['steel tube', (32.34e-03, 60)],
43             ['steel tube', (32.34e-03, 180)],
44             ['steel tube', (32.34e-03, 300)]]
45
46 vs = np.array([[1], [a ** 2], [a], [0], [0], [0]]) * V
47 Rs = Ls = np.zeros((6, 6))
48 vl = np.zeros((6, 1))
49 Rl = np.diag([60, 60, 60, 0, 0, 0])
50 Ll = np.diag([0.3, 0.3, 0.3, 0, 0, 0])
51
52 cable_length = 1
53
54 umb = system.Umbilical(elements, cable_length)
55 source_inst = system.Source(vs, Rs, Ls)
56 load_inst = system.Load(vl, Rl, Ll)
57
58 z = 0
59 f_step = 20
60 f_array = np.arange(1, 5*10**3 + f_step, f_step)
61
62 in_data = np.loadtxt('ex_B_R_L_marius.txt', skiprows=1)
63 f_data = np.loadtxt('ex_B_10_km_duplex_voltage.txt')
64 R_m, L_m = in_data[:, 0], in_data[:, 1]
65 f_array_m = f_data[:, 0]
66
67
68 impedance = solver.solution_single_z(f_array, z, umb, source_inst, load_inst,
69                                       'z')[:, 0]
70 R = impedance.real * 10 ** 3
71 X = impedance.imag
72 L = X / (2 * math.pi * f_array) * 10 ** 6
73
74
75 def z_in(f, r, l):
76     w = 2 * math.pi * f
77     c = 1.7080454e-10
78
79     gamma = cmath.sqrt((r + 1j * w * l) * 1j * w * c)
80     Z_c = cmath.sqrt((r + 1j * w * l) / (1j * w * c))
81     Z_L = 60 + 1j * w * 0.3
```

```
82      z_in = Z_c * ((Z_L + Z_c * cmath.tanh(gamma * cable_length)) /
83                    (Z_c + Z_L * cmath.tanh(gamma * cable_length)))
84      return abs(z_in)
85
86
87  zinvec = np.absolute(np.vectorize(z_in)(f_array_m, R_m / 10**3, L_m / 10**6))
88  plt.plot(f_array_m, zinvec, 'ko', ms=9, linewidth=0, label='Flux2D')
89  plt.plot(f_array, np.absolute(impedance), 'k', linewidth=3, label='UmbSIm')
90  plt.xlabel('Frequency [Hz]')
91  plt.ylabel(r'Impedance magnitude [$\Omega$]')
92  plt.ylim([0, 950])
93  plt.yticks([0, 200, 400, 600, 800, 950])
94  plt.xlim([1, 5.1e03])
95  plt.legend()
96  plt.show()
97
98  plt.plot(f_array, R, color='k', linewidth=3, label='UmbSim')
99  plt.plot(f_array_m, R_m, color='k', linewidth=3, linestyle='--', label='Flux2D')
100 plt.xlabel('Frequency [Hz]')
101 plt.ylabel(r'Resistance [$\Omega$/km]')
102 plt.xlim([0, 5.1e03])
103 plt.legend(loc=2)
104 plt.grid(True)
105 plt.show()
106
107 plt.plot(f_array, L, color='k', linewidth=3, label='UmbSim')
108 plt.plot(f_array_m, L_m, color='k', linewidth=3, linestyle='--', label='Flux2D')
109 plt.xlabel('Frequency [Hz]')
110 plt.ylabel(r'Inductance [mH/km]')
111 plt.xlim([0, 5.1e03])
112 plt.legend()
113 plt.grid(True)
114 plt.show()
```

C.3 Example A3

```
# -*- coding: utf-8 -*-

"""
Example A3 - Harmonic Analysis

Harmonic analysis of Umbilical A2 with voltage source as defined from the limits
of the harmonic spectrum as recommended in the IEC 61000-2-4 Class 2 standard.
"""

__author__ = 'Martin Hovde'
__email__ = 'martin.hovde@nmbu.no'

import math, cmath
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from UmbSim import system, solver
matplotlib.rcParams.update({'font.size': 24})

system.PowerPhase.set_parameters({'d_core': 11.5e-03,
                                  'd_inner_semi_con': 13.5e-03,
                                  'd_core_insulation': 29.5e-03,
                                  'd_outer_semi_con': 37.9e-03,
                                  'core_dc_resistance': 1.93e-04,
                                  'screen': False})

system.SteelTube.set_parameters({'d_inner': 1.27e-02,
                                 'd_outer': 1.562e-02,
                                 'd_sheath': 1.902e-02,
                                 'rho_tube': 8e-07})

system.Pipe.set_parameters({'d_pipe': 189e-03,
                            'rho_pipe': 0.3,
                            'mu_r_p': 1})

elements = [['power phase', (22.3e-03, 0)],
            ['power phase', (22.3e-03, 120)],
            ['power phase', (22.3e-03, 240)],
            ['steel tube', (32.34e-03, 60)],
            ['steel tube', (32.34e-03, 180)],
            ['steel tube', (32.34e-03, 300)]]

a = cmath.exp(1j * 2 * math.pi / 3)
V = 36 * 10 ** 3 / math.sqrt(3)
n = 6  # Number of conductors
vs = np.array([1, a ** 2, a, 0, 0, 0]).transpose() * V
Rs = Ls = np.zeros((n, n))
vl = np.zeros((n, 1))
Rl = np.diag([60., 60., 60., 0, 0, 0])
Ll = np.diag([0.3, 0.3, 0.3, 0, 0, 0])

cable_length = 31e03

umb = system.Umbilical(elements, cable_length)
source_inst = system.Source(vs, Rs, Ls)
load_inst = system.Load(vl, Rl, Ll)


def x_1(h):
    return abs((2.27 * (17. / h) - 0.27)) / 100


def x_2(h):
    return abs((0.25 * (10. / h) + 0.25)) / 100


def harmonic_voltage_list():
```

```
68      h_list = [False, 1, 0.02, 0.05, 0.01, 0.06, 0.005, 0.05, 0.005,
69                0.015, 0.005, 0.035, False, 0.03, False, 0.004, False,
70                0.02, False, False, False, 0.003]
71      voltage_list = []
72      for h in range(1, 51):
73          if h <= 21:
74              if h_list[h]:
75                  voltage_list.append(h_list[h])
76              elif not h % 3 and h % 2:
77                  # Triplen
78                  voltage_list.append(0.002)
79              elif not h % 2:
80                  # Even
81                  voltage_list.append(x_2(h))
82              else:
83                  # Odd non-triplen
84                  voltage_list.append(x_1(h))
85          elif h > 21:
86              if not h % 3 and h % 2:
87                  # Triplen
88                  voltage_list.append(0.002)
89              elif not h % 2:
90                  # Even
91                  voltage_list.append(x_2(h))
92              else:
93                  # Odd non-triplen
94                  voltage_list.append(x_1(h))
95      return voltage_list
96
97  v_harmonic = np.array(harmonic_voltage_list()) * 100
98  THD = math.sqrt(sum(v_harmonic[1:] ** 2))
99  print 'Voltage total harmonic distortion is', THD, '% at the sending end.'
100
101
102 plt.bar(range(2, 51), v_harmonic[1:], 0.3, color='k', align='center')
103 plt.xlim([1, 51])
104 plt.xticks(np.arange(2, 51, 4))
105 plt.ylim([0, 7])
106 plt.xlabel(r'Harmonic order $h$')
107 plt.ylabel(r'$\mathrm{v}_h$ as percentage of fundamental [%]')
108 plt.show()
109
110 f_fundamental = 50
111 f_harmonic = np.arange(1, 51) * f_fundamental
112 z_array = np.linspace(0, cable_length, 300)
113
114 v_sqrd = np.zeros((300, 6), dtype=np.complex_)
115 i_sqrd = np.zeros((300, 6), dtype=np.complex_)
116 v_source_sqrd = 0
117
118 for h, f in enumerate(f_harmonic):
119     vs = np.array([1, a ** (2 * (h+1)), a ** (h+1), 0, 0, 0]).transpose() * \
120          V * v_harmonic[h] / 100
121     v_source_sqrd += vs[0] ** 2
122     load_inst = system.Load(vl, Rl, Ll)
123     source_inst = system.Source(vs, Rs, Ls)
124     vh = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
125                                               load_inst, 'v'))
126     ih = np.absolute(solver.solution_single_f(f, z_array, umb, source_inst,
127                                               load_inst, 'i'))
128
129     v_sqrd += vh ** 2
130     i_sqrd += ih ** 2
131
132
133 v = np.sqrt(v_sqrd)
134 i = np.sqrt(i_sqrd)
135 v_source = np.sqrt(v_source_sqrd)
136 print 'The RMS source voltage due to harmonic content is', \
137     abs(v_source / 10**3), 'kV.'
```

```
138 print 'which is', abs(v_source - V) / V * 100, \
139     '% higher than the fundamental source voltage.'
140
141 vs = np.array([[1], [a ** 2], [a], [0], [0], [0]]) * V
142 source_inst = system.Source(vs, Rs, Ls)
143 v_1 = solver.solution_single_f(50, z_array, umb, source_inst, load_inst,
144                                'v')
145 i_1 = solver.solution_single_f(50, z_array, umb, source_inst, load_inst,
146                                'i')
147
148 matplotlib.rcParams.update({'font.size': 24})
149
150 plt.plot(z_array / 10**3, abs(v[:, 3]), color='k', linewidth=4,
151          label=r'$\sqrt{v_1^2 + \sum_{h=2}^{50} v_h^2}$')
152 plt.plot(z_array / 10**3, abs(v_1[:, 3]), color='k', linestyle='--',
153          linewidth=3, label='Fundamental')
154 plt.xlabel('z [km]')
155 plt.ylabel('Voltage magnitude [V]')
156 plt.xlim([0, cable_length / 10**3])
157 plt.legend(loc=2)
158 plt.show()
159
160 plt.plot(z_array / 10**3, abs(v[:, 0] / 10**3), color='k', linewidth=3,
161          label=r'$\sqrt{v_1^2 + \sum_{h=2}^{50} v_h^2}$')
162 plt.plot(z_array / 10**3, abs(v_1[:, 0] / 10**3), color='k', linestyle='--',
163          linewidth=3, label='Fundamental')
164 plt.xlabel('z [km]')
165 plt.ylabel('Voltage magnitude [kV]')
166 plt.xlim([0, cable_length / 10**3])
167 plt.legend()
168 plt.show()
169
170 plt.plot(z_array / 10**3, abs(i[:, 3]), color='k', linewidth=4,
171          label=r'$\sqrt{i_1^2 + \sum_{h=2}^{50} i_h^2}$')
172 plt.plot(z_array / 10**3, abs(i_1[:, 3]), color='k', linestyle='--',
173          linewidth=3, label='Fundamental')
174 plt.xlabel('z [km]')
175 plt.ylabel('Current magnitude [A]')
176 plt.xlim([0, cable_length / 10**3])
177 plt.legend(loc=1)
178 plt.show()
179
180 plt.plot(z_array / 10**3, abs(i[:, 0]), color='k', linewidth=3,
181          label=r'$\sqrt{i_1^2 + \sum_{h=2}^{50} i_h^2}$')
182 plt.plot(z_array / 10**3, abs(i_1[:, 0]), color='k', linestyle='--',
183          linewidth=3, label='Fundamental')
184 plt.xlabel('z [km]')
185 plt.ylabel('Current magnitude [A]')
186 plt.xlim([0, cable_length / 10**3])
187 plt.legend(loc=4)
188 plt.show()
```

C.4 Example B1 and B2

```
# -*- coding: utf-8 -*-

"""
Example B1 and B2
Script for:
* Input impedance simulation of Umbilical B1 and B2
* Comparison with frequency sweep measurements (in air and best fit)
"""

__author__ = 'Martin Hovde'
__email__ = 'martin.hovde@nmbu.no'

import cmath, math
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from UmbSim import system, solver
matplotlib.rcParams.update({'font.size': 24})

inner_circuit = False

if inner_circuit:
    # Inner circuit
    system.PowerPhase.set_parameters({'d_core': 10.9e-03,
                                      'd_inner_semi_con': 12.9e-03,
                                      'd_core_insulation': 24.2e-03,
                                      'd_outer_semi_con': 27.2e-03,
                                      'core_dc_resistance': 1.93e-04,
                                      'screen': False})
else:
    # Outer circuit
    system.PowerPhase.set_parameters({'d_core': 7.7e-03,
                                      'd_inner_semi_con': 9.7e-03,
                                      'd_core_insulation': 16.8e-03,
                                      'd_outer_semi_con': 19.8e-03,
                                      'core_dc_resistance': 3.87e-04,
                                      'screen': False})

system.SteelTube.set_parameters({'d_inner': 19.05e-03,
                                 'd_outer': 21.65e-03,
                                 'd_sheath': 25.7e-02,
                                 'rho_tube': 8e-07})

# rho air: 1.3e14
# rho armour: 2e-7
# best fit mu_r: 15

system.Pipe.set_parameters({'d_pipe': 136e-03,
                            'rho_pipe': 2e-7,
                            'mu_r_p': 15})

a = cmath.exp(1j * 2 * math.pi / 3)
if inner_circuit:
    V = 12e03 / math.sqrt(3)
    elements = [['power phase', (18.5e-03, 0)],
                ['power phase', (18.5e-03, 120)],
                ['power phase', (18.5e-03, 240)]]

    vs = np.array([1, a ** 2, a]).transpose() * V
    n = 3
    Rs = Ls = np.zeros((n, n))
    vl = np.zeros((n, 1))
    Rl = np.diag([1.0e10, 1.0e10, 1.0e10])
    Ll = np.zeros((n, n))
else:
    V = 24e03 / math.sqrt(3)
    elements = [['power phase', (47.5e-03, 0)],
```

```
68                  ['power phase', (47.5e-03, 120)],
69                  ['power phase', (47.5e-03, 240)],
70                  ['steel tube', (47.5e-03, 32)],
71                  ['steel tube', (47.5e-03, 152)]]
72
73      vs = np.array([1, a ** 2, a, 0, 0]).transpose() * V
74      n = 5
75      Rs = np.diag([0, 0, 0, 1e07, 1e07])
76      Ls = np.zeros((n, n))
77      vl = np.zeros((n, 1))
78      Rl = np.diag(np.repeat(1e07, 5))
79      Ll = np.zeros((n, n))
80
81  cable_length = 42e03
82
83  umb = system.Umbilical(elements, cable_length)
84  source_inst = system.Source(vs, Rs, Ls)
85  load_inst = system.Load(vl, Rl, Ll)
86
87  z = 0
88  f_step = 15
89  f_array = np.arange(0.1, 5*10**3 + f_step, f_step)
90  z_array = np.linspace(0, cable_length, 50)
91
92  impedance = solver.solution_single_z(f_array, z, umb, source_inst, load_inst,
93                                       'z')
94
95  in_data_L1 = np.loadtxt('L1.txt', skiprows=1)
96  in_data_L2 = np.loadtxt('L2.txt', skiprows=1)
97  in_data_L3 = np.loadtxt('L3.txt', skiprows=1)
98  in_data_inner = np.loadtxt('Inner.txt', skiprows=1)
99  in_data_angle = np.loadtxt('Inner_angle.txt', skiprows=1)
100
101 f_array_L1, impedance_L1 = in_data_L1[:, 0], in_data_L1[:, 1]
102 f_array_L2, impedance_L2 = in_data_L2[:, 0], in_data_L2[:, 1]
103 f_array_L3, impedance_L3 = in_data_L3[:, 0], in_data_L3[:, 1]
104 f_array_inner, impedance_inner = in_data_inner[:, 0], in_data_inner[:, 1]
105 f_array_angle, impedance_angle = in_data_angle[:, 0], in_data_angle[:, 1]
106
107 if inner_circuit:
108     plt.plot(f_array_inner, impedance_inner, 'ko', ms=9, linewidth=0,
109              label='Measured')
110     plt.plot(f_array, np.absolute(impedance[:, 0]), color='k', linewidth=3,
111              label='UmbSim')
112     plt.xlim([0, 5.05e03])
113     plt.ylim([0, 300])
114     plt.ylabel(r'Impedance magnitude [$\Omega$]')
115     plt.xlabel('Frequency [Hz]')
116     plt.legend()
117     plt.show()
118
119     plt.plot(f_array_angle, impedance_angle, 'ko', ms=9, linewidth=0,
120              label='Measured')
121     plt.plot(f_array, np.angle(impedance[:, 0], deg=True), color='k',
122              linewidth=3, label='UmbSim')
123     plt.xlim([0, 5.05e03])
124     plt.ylim([-92, 92])
125     plt.yticks([-90, -45, 0, 45, 90])
126     plt.ylabel(r'Impedance angle [deg $\degree$]')
127     plt.xlabel('Frequency [Hz]')
128     plt.legend()
129     plt.show()
130
131 else:
132     # L3 impedance
133     plt.plot(f_array_L1, impedance_L1, 'ko', ms=9, linewidth=0,
134              label='Measured')
135     plt.plot(f_array, np.absolute(impedance[:, 2]), color='k', linewidth=3,
136              label='UmbSim')
137     plt.xlim([0, 5.05e03])
```

```
138     plt.ylim([0, 300])
139     plt.ylabel(r'Impedance magnitude [$\Omega$]')
140     plt.xlabel('Frequency [Hz]')
141     plt.legend()
142     plt.show()
143
144     plt.plot(f_array, np.angle(impedance[:, 2], deg=True), color='k',
145              linewidth=3, label='L3 UmbSim')
146     plt.xlim([0, 5.05e03])
147     plt.ylim([-92, 92])
148     plt.yticks([-90, -45, 0, 45, 90])
149     plt.ylabel(r'Impedance angle [deg $\degree$]')
150     plt.xlabel('Frequency [Hz]')
151     plt.show()
152
153     # L2 impedance
154     plt.plot(f_array_L3, impedance_L3, 'ko', ms=9, linewidth=0,
155              label='Measured')
156     plt.plot(f_array, np.absolute(impedance[:, 1]), color='k', linewidth=3,
157              label='UmbSim')
158     plt.xlim([0, 5.05e03])
159     plt.ylim([0, 300])
160     plt.ylabel(r'Impedance magnitude [$\Omega$]')
161     plt.xlabel('Frequency [Hz]')
162     plt.legend()
163     plt.show()
164
165     plt.plot(f_array, np.angle(impedance[:, 1], deg=True), color='k',
166              linewidth=3, label='L3 UmbSim')
167     plt.xlim([0, 5.05e03])
168     plt.ylim([-92, 92])
169     plt.yticks([-90, -45, 0, 45, 90])
170     plt.ylabel(r'Impedance angle [deg $\degree$]')
171     plt.xlabel('Frequency [Hz]')
172     plt.show()
173
174     # L1 impedance
175     plt.plot(f_array_L2, impedance_L2, 'ko', ms=9, linewidth=0,
176              label='Measured')
177     plt.plot(f_array, np.absolute(impedance[:, 0]), color='k', linewidth=3,
178              label='UmbSim')
179     plt.xlim([0, 5.05e03])
180     plt.ylim([0, 300])
181     plt.ylabel(r'Impedance magnitude [$\Omega$]')
182     plt.xlabel('Frequency [Hz]')
183     plt.legend()
184     plt.show()
185
186     plt.plot(f_array, np.angle(impedance[:, 0], deg=True), color='k',
187              linewidth=3, label='L3 UmbSim')
188     plt.xlim([0, 5.05e03])
189     plt.ylim([-92, 92])
190     plt.yticks([-90, -45, 0, 45, 90])
191     plt.ylabel(r'Impedance angle [deg $\degree$]')
192     plt.xlabel('Frequency [Hz]')
193     plt.show()
```

D. Data from Flux2D simulations

Example A1 - Input impedance magnitude

Frequency [Hz]	Impedance [Ohm]		
50.0	139.36775320766677	2575.510204081633	21.0818822863667
151.0204081632653	551.9703347939185	2676.530612244898	12.359812906751493
252.0408163265306	152.94121392485675	2777.551020408163	8.26971984244208
353.0612244897959	87.93286799579393	2878.571428571428	11.5777803142151
454.0816326530612	58.32623624669673	2979.591836734694	18.222712568793288
555.1020408163265	39.9767621474468	3080.612244897959	25.945001590819075
656.1224489795918	26.692993571331428	3181.632653061225	34.62696704229518
757.1428571428571	16.155287336914025	3282.65306122449	44.73568626626895
858.1632653061224	7.6945909444820195	3383.673469387755	57.19967467434129
959.1836734693877	5.585224412697793	3484.69387755102	73.76689588228275
1060.204081632653	12.33617164933038	3585.714285714286	98.00506121483431
1161.224489795918	20.783590451909078	3686.734693877551	137.74714415635208
1262.244897959184	30.295704409089474	3787.755102040816	204.9870411707839
1363.265306122449	41.45564906045452	3888.775510204082	239.33242325838035
1464.285714285714	55.43948432717584	3989.795918367347	157.32573292897058
1565.30612244898	74.54691217301325	4090.816326530612	94.79024912617467
1666.326530612245	103.90646924636619	4191.836734693878	60.08389646824316
1767.34693877551	157.1714770331526	4292.857142857143	38.776197839833394
1868.367346938775	272.74193765947973	4393.877551020408	24.465946689965275
1969.387755102041	344.91633602538025	4494.897959183673	14.971208895579698
2070.408163265306	189.43985388632734	4595.918367346939	11.268794933449277
2171.428571428572	110.66768843777275	4696.938775510204	14.24563107528464
2272.448979591837	71.85268827445401	4797.959183673469	20.29360186689406
2373.469387755102	48.658502662857835	4898.979591836735	27.36359611006874
2474.489795918367	32.82068534071507	5000.0	35.17877551828574

Example A2 - Input impedance magnitude

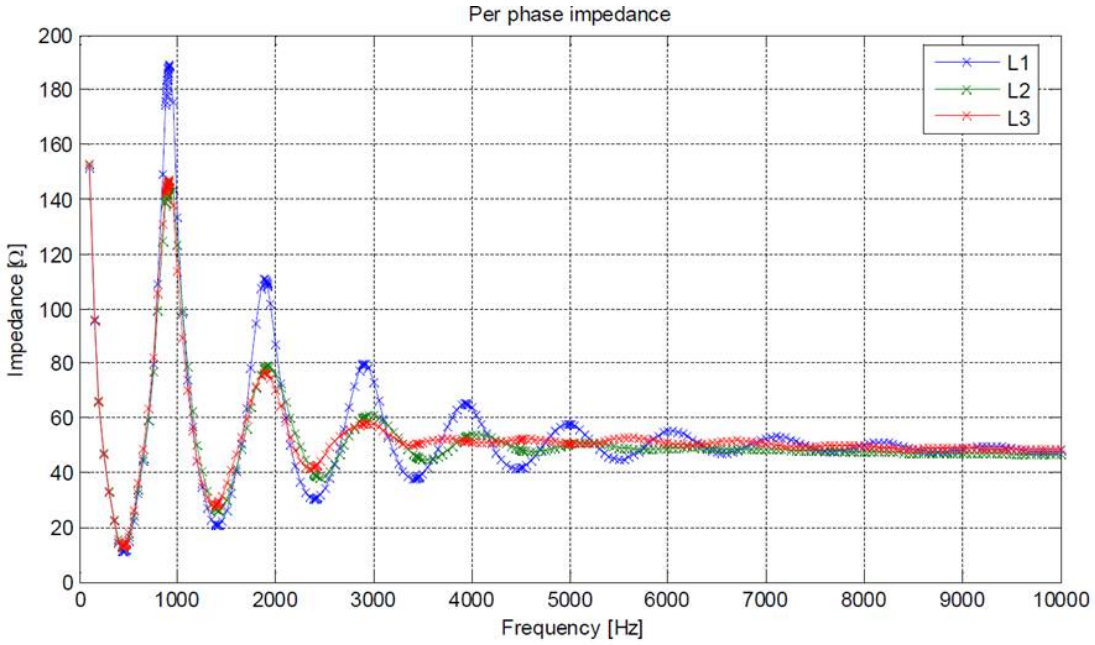
Frequency [Hz]	Impedance [Ohm]
50.0	139.53915948276466
151.0204081632653	551.7126711255376
252.0408163265306	152.6730692090333
353.0612244897959	87.51432306668777
454.0816326530612	57.76411382954133
555.1020408163265	39.27905246965687
656.1224489795918	25.876577687800474
757.1428571428571	15.285861585124074
858.1632653061224	7.198929408954162
959.1836734693877	7.0729072696565405
1060.204081632653	14.3767167002659
1161.224489795918	23.179934246038158
1262.244897959184	33.16510137742479
1363.265306122449	44.99989472028993
1464.285714285714	59.995897774972306
1565.30612244898	80.70093177315
1666.326530612245	112.59187208534213
1767.34693877551	168.06113120321982
1868.367346938775	255.7896789449046
1969.387755102041	243.41792921136835
2070.408163265306	150.71433827268933
2171.428571428572	96.08558262930543
2272.448979591837	65.03132488521153
2373.469387755102	45.13910041760394
2474.489795918367	31.178906878171095
2575.510204081633	21.00769945957011
2676.530612244898	14.321336495495773
2777.551020408163	12.550842687661762
2878.571428571428	15.909294199827011
2979.591836734694	21.90784112194882
3080.612244897959	29.1852181916912
3181.632653061225	37.5642318300734
3282.65306122449	47.37617727660693
3383.673469387755	59.326147868495404
3484.69387755102	74.5929695611362
3585.714285714286	94.9216210535999
3686.734693877551	121.56653544156195
3787.755102040816	147.68211172272012
3888.775510204082	146.5487202946855
3989.795918367347	114.22166363521421
4090.816326530612	80.95529938056978
4191.836734693878	56.81937049644893
4292.857142857143	40.118689973575655
4393.877551020408	28.692118868293015
4494.897959183673	21.662072792979888
4595.918367346939	19.09977490138983
4696.938775510204	20.62356013963441
4797.959183673469	24.793046612402396
4898.979591836735	30.415062276226806
5000.0	37.01926932914991

Example A2 - Resistance and inductance

R [Ohm/km]	L [mH/km]
0.193361049	0.430792423
0.199158838	0.429852062
0.209888104	0.428225141
0.224394799	0.426212778
0.241706248	0.424021664
0.261079982	0.421769483
0.281943076	0.419524644
0.303853727	0.417330399
0.326483071	0.415214616
0.349599559	0.413193815
0.373050409	0.411275723
0.396741991	0.409461545
0.420621863	0.407748012
0.444664035	0.406129091
0.468857777	0.404597288
0.493200011	0.40314458
0.517690102	0.401763023
0.54232689	0.400445125
0.567106836	0.399184044
0.592023595	0.397973672
0.617067527	0.396808644
0.642226152	0.395684307
0.667484397	0.394596659
0.692825331	0.393542284
0.718230282	0.392518279
0.743679883	0.391522187
0.769153651	0.39055193
0.794631537	0.389605758
0.820093222	0.388682189
0.845518771	0.387779974
0.870889275	0.386898052
0.896186298	0.386035521
0.921392403	0.385191607
0.946491388	0.384365646
0.971468166	0.383557058
0.996308515	0.382765334
1.020999439	0.38199002
1.045529701	0.381230711
1.069888679	0.380487037
1.094066804	0.379758653
1.118056154	0.379045242
1.141849175	0.378346498
1.165439645	0.377662133
1.188822845	0.376991869
1.211993889	0.376335432
1.234949155	0.375692556
1.257686587	0.375062983
1.280203724	0.374446451
1.302499034	0.373842705
1.32457184	0.373251491

E. Data from measurements

Example B1



Example B2

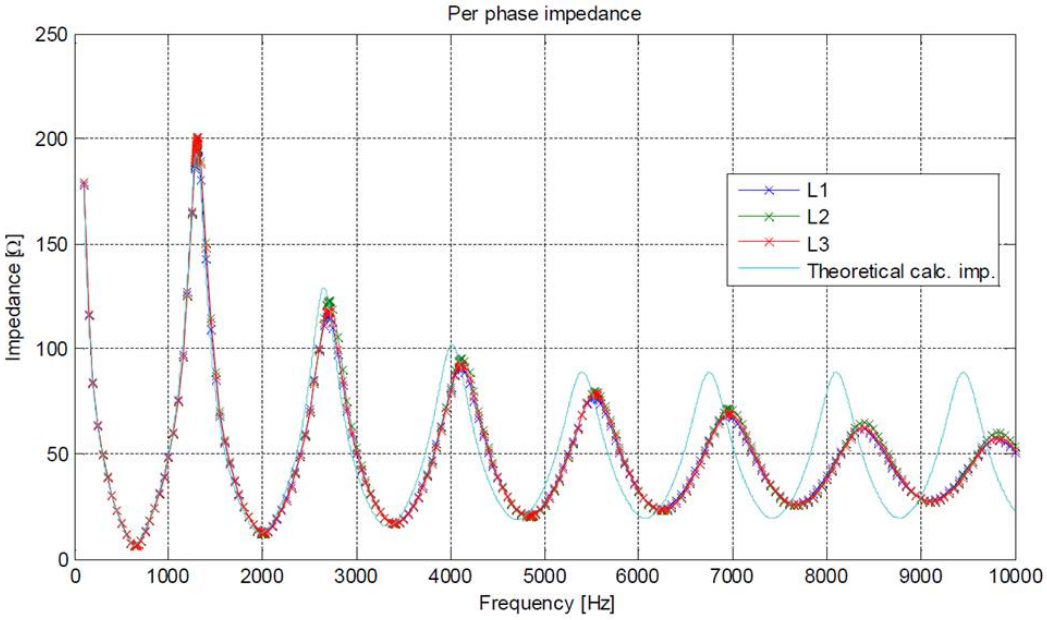


Figure 12 Measured per phase impedance as function of frequency

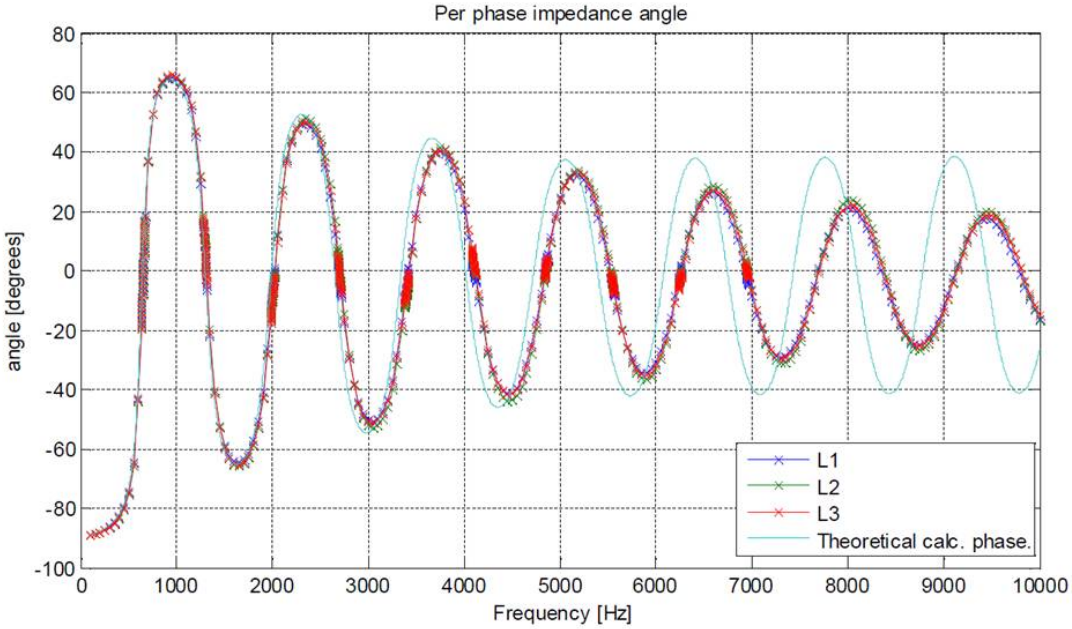
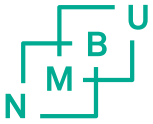


Figure 13 Measured per phase impedance angle as function of frequency



Norges miljø- og biovitenskapelig universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway