Steady State Simulations of a Hybrid HVAC/HVDC Network Using OS Based ARM Devices

Ioan Catalin Damian, Mircea Eremia
UNIVERSITY POLITEHNICA OF BUCHAREST
Splaiul Independenței 313, Sector 6
Bucharest, Romania
E-Mail: ioan_catalin.damian@upb.ro, eremia1@yahoo.com
URL: http://www.dsee.upb.ro

Keywords

«AC-DC converter», «Multi-terminal HVDC», «Hybrid simulation».

Abstract

For steady state calculations, all approaches make use of established computing devices (mostly PCs with x86-64 or x86-32 architecture) and environments that offer excellent performance but at a high cost. This paper presents an innovative alternative that is based on emerging technologies which have an ARM architecture, are still able to run powerful software, yet meet cost requirements that are less than 60 USD.

Introduction

The past 20 years have revealed a growing need to transmit large amounts of power over longer distances (greater than 500 km). This has made high voltage DC (HVDC) networks extremely enticing, considering numerous advantages over high voltage AC (HVAC) transmission, such as: lower losses, greater controllability, lower overhead/underground line footprint and possibility to use existing HVAC pillars. While development of HVDC started with point-to-point links, it continued with multiterminal networks (e.g. Nan'ao project in 2013, Zhoushan in 2014, ZhangBei in 2020 etc.). This has prompted the need to grow algorithms that can be used to perform steady-state calculations for systems that have both HVAC and HVDC. Presently, there are two major categories of steady-state algorithms: a unified method and a sequential method [1,2,3].

The unified approach evaluates the complete system model and performs all calculations in one sweep. The sequential method alternates the calculation process between the AC region and the DC region, with the use of established iterative methods such as Seidel-Gauss, Newton-Raphson and other advanced methods [2]. Because in the sequential method the modified zone is treated only in the perspective of admittance matrix alterations and does not imply algorithm changes in the remaining zones, this approach will be exploited in the following sections.

The computing tools that enable streamlined simulations for steady-state analysis are mostly based on x86-64 or x86-32 PC architectures and have greatly over time. Moreover, computing performance has become so great that it takes only tens of seconds to simulate highly complex networks. However, these computing instruments cost at least 500 USD, without taking into consideration licensing fees related to advanced software packages. The Advanced RISC Machines (ARM) architecture was first proposed in the late 1980s, but it gained substantial traction after 2000, propelled by the expanding market of smartphones. However, in 2012, the Raspberry Pi Foundation launched the first single-board computer based on the ARM architecture. This device reached consumer acclaim for its versatility and low cost and paved the way for an accelerated development of automation projects such as: robot controllers, smart home hubs, factory controller etc. [4]. The current iteration of this device (version 4B) is sufficiently powerful to run even demanding operating systems such as Windows 10. However, no one has attempted to perform complex HVAC/HVDC network steady state simulations on such a device. This paper approaches this topic, using mathematical models that are detailed in the following

paragraphs and are implemented using the C# programming language on a PC and on a Raspberry Pi 4, both of them running Windows 10 natively.

Steady State Model of a Hybrid HVAC/HVDC Network

A steady state model of a hybrid network consists of three zones of interest. One zone corresponds to the AC network, one relates to the DC network and lastly a zone that reflects the interface between the AC and DC regions. PCC represents the point of common coupling, between the AC system and the HVDC converter station.

Considering that a sequential method is used to perform steady state calculations, this leads to an alternation between the AC zone and the DC zone.

AC System Calculations

The AC region calculations can be performed using well developed techniques, such as Newton-Raphson. In this method, the goal is to iteratively adjust parameters using a Jacobian matrix, until a convergence condition is reached [2,3,5].

For each *PQ* node, the following powers are calculated:

$$\begin{aligned} P_i^{imp} &= P_{g,i} - P_{c,i} \\ Q_i^{imp} &= Q_{g,i} - Q_{c,i} \end{aligned} \tag{1}$$

$P_{g,i}$ and $Q_{g,i}$ are the active and reactive generated powers (at node *i*), $P_{c,i}$ and $Q_{c,i}$ are the active and reactive consumed powers (at node *i*).

In the first iteration, node voltages ($U_i^{(0)}$) are initialized using 1 p.u. value and the phase angles ($\theta_i^{(0)}$) are set at 0.

Looking at the *PU* nodes, the following powers are calculated:

$$\begin{aligned} P_i^{imp} &= P_{g,i} - P_{c,i} \\ Q_i^{\min} &= Q_{g,i}^{\min} - Q_{c,i} \\ Q_i^{\max} &= Q_{g,i}^{\max} - Q_{c,i} \end{aligned} \tag{2}$$

Q_i^{min} and Q_i^{max} are the maximum and minimum reactive powers (at node i).

The slack bus nodes are regarded only for voltage calculation, while the angle remains 0:

$$U_i = U_i^{imp};\ \theta_i = 0$$

In the following step, it is necessary to calculate node powers, as shown in (3):

$$\begin{aligned} P_i^{(p)} &= real\left(\underline{S}_i^{(p)}\right) = \sum_{k=1}^{n} U_i^{(p)} U_k^{(p)} \left[G_{ik}\, cos\left(\theta_i^{(p)} - \theta_k^{(p)}\right) + B_{ik}\, sin\left(\theta_i^{(p)} - \theta_k^{(p)}\right)\right] \\ Q_i^{(p)} &= imag\left(\underline{S}_i^{(p)}\right) = \sum_{k=1}^{n} U_i^{(p)} U_k^{(p)} \left[G_{ik}\, sin\left(\theta_i^{(p)} - \theta_k^{(p)}\right) - B_{ik}\, cos\left(\theta_i^{(p)} - \theta_k^{(p)}\right)\right] \end{aligned} \tag{3}$$

If the iteration number *p* is greater than 2, for all *PU* nodes, the reactive power confinement limits must be verified:

- If $Q_i^{(p)} < Q_i^{min}$, then the node becomes *PQ*, having $Q_i^{imp} = Q_i^{min}$;
- If $Q_i^{(p)} > Q_i^{min}$, then the node becomes *PQ*, having $Q_i^{imp} = Q_i^{max}$.

Next, for *PU* and *QP* nodes, power corrections are evaluated:

- If the node is *PU*, then:

$$\Delta P_i^{(p)} = P_i^{imp} - P_i^{(p)} \tag{4.a}$$

- If the node is *PQ*, then:

$$\Delta P_i^{(p)} = P_i^{imp} - P_i^{(p)} \tag{4.b}$$

$$\Delta Q_i^{(p)} = Q_i^{imp} - Q_i^{(p)} \tag{4.c}$$

The convergence test is performed using (5.*a*) and (5.*b*).

$$\max_i \left\{ \left| \Delta P_i^{(p)} \right| \right\} \leq \varepsilon \tag{5.a}$$

$$\max_i \left\{ \left| \Delta Q_i^{(p)} \right| \right\} \leq \varepsilon \tag{5.b}$$

If the convergence check shows that the condition is valid, the Newton-Raphson procedure can end with grid power-flow calculations and DC network calculations can commence. However, if the established threshold is bigger than the power corrections, the algorithm must continue with the calculation of the Jacobian matrix ($J^{(p)}$) as in (6), using the terms defined in (7.*a*) and (7.*b*).

$$\mathbf{J}^{(p)} = \begin{array}{|c|c|} \hline \mathbf{H}^{(p)} & \mathbf{K}^{(p)} \\ \hline \mathbf{M}^{(p)} & \mathbf{L}^{(p)} \\ \hline \end{array} \tag{6}$$

$$H = \begin{bmatrix} \frac{\partial P_1}{\partial \theta_1} & \cdots & \frac{\partial P_1}{\partial \theta_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial P_N}{\partial \theta_1} & \cdots & \frac{\partial P_N}{\partial \theta_N} \end{bmatrix} \qquad K = \begin{bmatrix} \frac{\partial P_1}{\partial U_1} U_1 & \cdots & \frac{\partial P_1}{\partial U_{n_c}} U_{n_c} \\ \vdots & \ddots & \vdots \\ \frac{\partial P_N}{\partial U_1} U_1 & \cdots & \frac{\partial P_N}{\partial U_{n_c}} U_{n_c} \end{bmatrix} \tag{7.a}$$

$$M = \begin{bmatrix} \frac{\partial Q_1}{\partial \theta_1} & \cdots & \frac{\partial Q_1}{\partial \theta_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial Q_{n_c}}{\partial \theta_1} & \cdots & \frac{\partial Q_{n_c}}{\partial \theta_N} \end{bmatrix} \qquad L = \begin{bmatrix} \frac{\partial Q_1}{\partial U_1} U_1 & \cdots & \frac{\partial Q_1}{\partial U_{n_c}} U_{n_c} \\ \vdots & \ddots & \vdots \\ \frac{\partial Q_{n_c}}{\partial U_1} U_1 & \cdots & \frac{\partial Q_{n_c}}{\partial U_{n_c}} U_{n_c} \end{bmatrix} \tag{7.b}$$

The corrections for the phase angle $\Delta\boldsymbol{\theta}^{(p)}$ and the relative voltage $\left(\frac{\Delta \mathbf{U}}{\mathbf{U}}\right)^{(p)}$ can be calculated using (8):

$$\begin{array}{|c|c|} \hline \mathbf{H}^{(p)} & \mathbf{K}^{(p)} \\ \hline \mathbf{M}^{(p)} & \mathbf{L}^{(p)} \\ \hline \end{array} \begin{array}{|c|} \hline \Delta\boldsymbol{\theta}^{(p)} \\ \hline \left(\frac{\Delta \mathbf{U}}{\mathbf{U}}\right)^{(p)} \\ \hline \end{array} = \begin{array}{|c|} \hline \Delta P^{(p)} \\ \hline \Delta Q^{(p)} \\ \hline \end{array} \tag{8}$$

With the previous corrections, it is now possible to determine the values of the phase angle and the node voltage for the following iteration:

$$\theta_i^{(p+1)} = \theta_i^{(p)} + \Delta\theta_i^{(p)} \tag{9}$$

- If the node is *PQ* then:

$$U_i^{(p+1)} = U_i^{(p)} \left(1 + \frac{\Delta U_i^{(p)}}{U_i^{(p)}} \right) \tag{10}$$

- If the node was initially of type *PU* but is now a *PQ* node:

- If $Q_i^{imp} = Q_i^{min}$ and $U_i^{(p+1)} < U_i^{imp}$ then $U_i^{(p+1)} = U_i^{imp}$ and the node returns to *PU* type
- Also, if $Q_i^{imp} = Q_i^{max}$ and $U_i^{(p+1)} > U_i^{imp}$ then $U_i^{(p+1)} = U_i^{imp}$ and the node returns to *PU* type

Next, the computation procedure continues with the next iteration, by following the same algorithm shown from (3) onwards.

DC System and Converter Interface Calculations

The DC system, with converter interface, can be approached using the Newton-Raphson iterative method [3,6,7,8,9].

In order to determine steady-state values inside the DC network (as well as for converter interface), the values previously determined for the AC region can now be used to initialize the computation process of the DC region. In the DC zone, as well as for the converter region (Figure 1), the Newton-Raphson method can again be used, with the help of several main equations. These are:

- The active and reactive power flow between the PCC and the AC system:

$$f_1 = P_s - \sum_{k=1}^{n} U_s U_k [G_{sk}\, cos(\theta_s - \theta_k) + B_{sk}\, sin(\theta_s - \theta_k)] \tag{11}$$

$$f_2 = Q_s - \sum_{k=1}^{n} U_s U_k [G_{sk}\, sin(\theta_s - \theta_k) - B_{sk}\, cos(\theta_s - \theta_k)] \tag{12}$$

where (*s*) is the PCC node and (*k*) is the AC node.

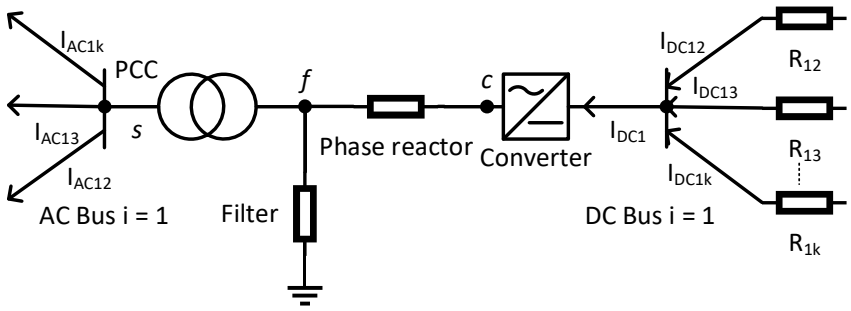


Fig. 1. AC/DC interface

- The power flow between the PCC node (*s*) and the converter (*c*):

$$f_3 = P_s - U_s\{-U_s G_{cs} + U_c[G_{cs}\, cos(\theta_s - \theta_c) + B_{cs}\, sin(\theta_s - \theta_c)]\} \tag{13}$$

$$f_4 = Q_s - U_s\{-U_s G_{cs} + U_c[G_{cs}\, sin(\theta_s - \theta_c) - B_{cs}\, cos(\theta_s - \theta_c)]\} \tag{14}$$

- The power balance between the AC side and the DC side of the converter (losses are neglected), leads to a 5th function:

$$f_5 = U_{DC} I_{DC} - U_c\{U_c G_{cs} - U_s[G_{cs}\, cos(\theta_c - \theta_s) + B_{cs}\, sin(\theta_c - \theta_s)]\} \tag{15}$$

- The current injection at the DC side,

$$f_6 = I_{DC,i} - \sum_{\substack{k=1 \\ k \neq i}}^{n} G_{ik}\, U_{DC,k} \tag{16}$$

where *i* is any DC side node.

As in the AC zone calculation procedure, it is necessary to define the Jacobian matrix. Afterwards, using power and voltage corrections, new values are calculated until the convergence threshold is reached.

Depending on the type of operational mode the converter is set, it is possible to specify additional functions:

1. Constant $P - Q$:

$$P_s = P_s^{imp} => f_7 = P_s - P_s^{imp} \quad (17.a)$$

$$Q_s = Q_s^{imp} => f_8 = Q_s - Q_s^{imp} \quad (18.a)$$

2. Constant $V_{DC} - Q$:

$$V_{DC} = V_{DC}^{imp} => f_7 = V_{DC} - V_{DC}^{imp} \quad (17.b)$$

$$Q_s = Q_s^{imp} => f_8 = Q_s - Q_s^{imp} \quad (18.b)$$

3. Constant $P - V_{AC}$:

$$P_s = P_s^{imp} => f_7 = P_s - P_s^{imp} \quad (17.c)$$

$$V_s = V_s^{imp} => f_8 = V_s - V_s^{imp} \quad (18.c)$$

4. Constant $V_{DC} - V_{AC}$:

$$V_{DC} = V_{DC}^{imp} => f_7 = V_{DC} - V_{DC}^{imp} \quad (17.d)$$

$$V_s = V_s^{imp} => f_8 = V_s - V_s^{imp} \quad (18.d)$$

The unknown variables, which can be determined using the Newton-Raphson method are identifiable from functions $f_1, f_2, \ldots f_8$: P_s, U_s, θ_s for f_1; Q_s, U_s, θ_s for f_2; P_s, U_s, U_c, θ_s, θ_c for f_3; Q_s, U_s, U_c, θ_s, θ_c for f_4; U_{DC}, I_{DC}, U_c, U_s, θ_s, θ_c for f_5; U_{DC}, I_{DC} for f_6; while f_7 and f_8 depend on the control strategy. The system is solvable since there are eight unknown variables and eight equations. These equations form the Jacobin matrix J. The corrections vector, ΔX, which has the structure in (9) can now be calculated.

$$\Delta X = [\Delta P_s, \Delta Q_s, \Delta U_s, \Delta \theta_s, \Delta U_c, \Delta \theta_c, \Delta U_{DC}, \Delta I_{DC}]^T \quad (19)$$

The correction vector adjusts the values of various electrical parameters, and the calculation process either continues with a new iteration or finishes, depending on the convergence criterion ε. If the DC network steady-state calculations reach the finish criterion, then all resulting values can be used to reinitialize the calculation process for the AC zone. The process repeats until a global convergence criterion is met.

Case Study

The aim of the case study is to demonstrate the validity of the concept that, using OS based ARM devices, it is possible to perform steady-state calculations for complex hybrid HVAC/HVDC networks, in an efficient and fast manner, that is comparable to the performance of expensive and highly advanced PCs.

Description of the Computing Setup

Simulation is performed on 2 types of devices. The first type consists of a PC with an Intel Core i7 1065G7, with 4 cores (8 logical processors), that has a maximum boost frequency of 3.9 GHz. The second device is a Raspberry Pi 4, that has a quad core Cortex A72 ARM CPU, running at 1.5 GHz. Both devices run Windows 10 natively.

The steady-state calculation program is developed using the C# programming language, in the Universal Windows Platform (UWP). Furthermore, there are three versions of the same program: one which targets x86-64, one which targets 32bit emulation on Raspberry Pi 4 and the other one which targets native ARM64 on Raspberry Pi 4.

The concept of running Windows 10 apps on a Raspberry Pi 4 is extremely new, and it is based on the "Windows on Raspberry Project" [9], in which open-source drivers are developed in order to facilitate the interface between the hardware layer and the operating system layer. With this approach, it is possible to run apps in two manners: deploying native apps that are compiled for 64bit ARM or using an emulation sandbox, that accompanies the operating system and enables compilation and deployment of x86-32 apps.

The steady-state software is designed to offer flexibility by allowing the user to change the global error margin as well as the number of consecutive runs. Additionally, the program is configured to calculate

the steady-state values using an error margin of 0.00003 s (AC zone error, DC zone error and global error). This value is chosen in order to increase computation effort and to test the limits of the computing equipment. Furthermore, to evaluate performance on the two different devices while also considering equipment-specific thermal limits, 10 consecutive runs are implemented.

Description of the Analyzed Hybrid Network

A complex HVAC/HVDC network is brought forward in order to test the performance of an ARM device. It consists of two onshore AC systems (system *A*, with nodes *A0* and *A1*, and system *B*, with nodes *B0*, *B1*, *B2* and *B3)*, 4 offshore AC systems (system *C*, with *C1* and *C2*, system *D*, with *D1*, system *E*, with *E1* and system *F*, with *F1),* 2 DC nodes which have no connection to the AC networks (*B4* and *B5*) and 3 DC systems (*DCS1*, with *A1* and *C1*, *DCS2*, with *B2*, *B3*, *B5*, *F1* and *E1*, and *DCS3*, with *A1*, *C2*, *D1*, *E1*, *B1*, *B4* and *B2*).

The entire hybrid network is shown in Figure 2, and it is based on a benchmark network [11] by *Cigre Working Group B4.57*, in which an additional line between nodes *Ba-A1* and *Ba-B1* has been added.

In the DC zone, there are symmetric monopole busses (identified with notation *Bm*) and bipole busses (*Bb*). There are also 2 DC-DC converters (*Cd-B1* and *Cd-E1*). The *DCS1* and *DCS2* systems have a voltage rating of ± 200 kV, while *DCS3* has a rated voltage of ± 400 kV. The AC onshore zones operate at 380 kV and the offshore zones operate at 145 kV.

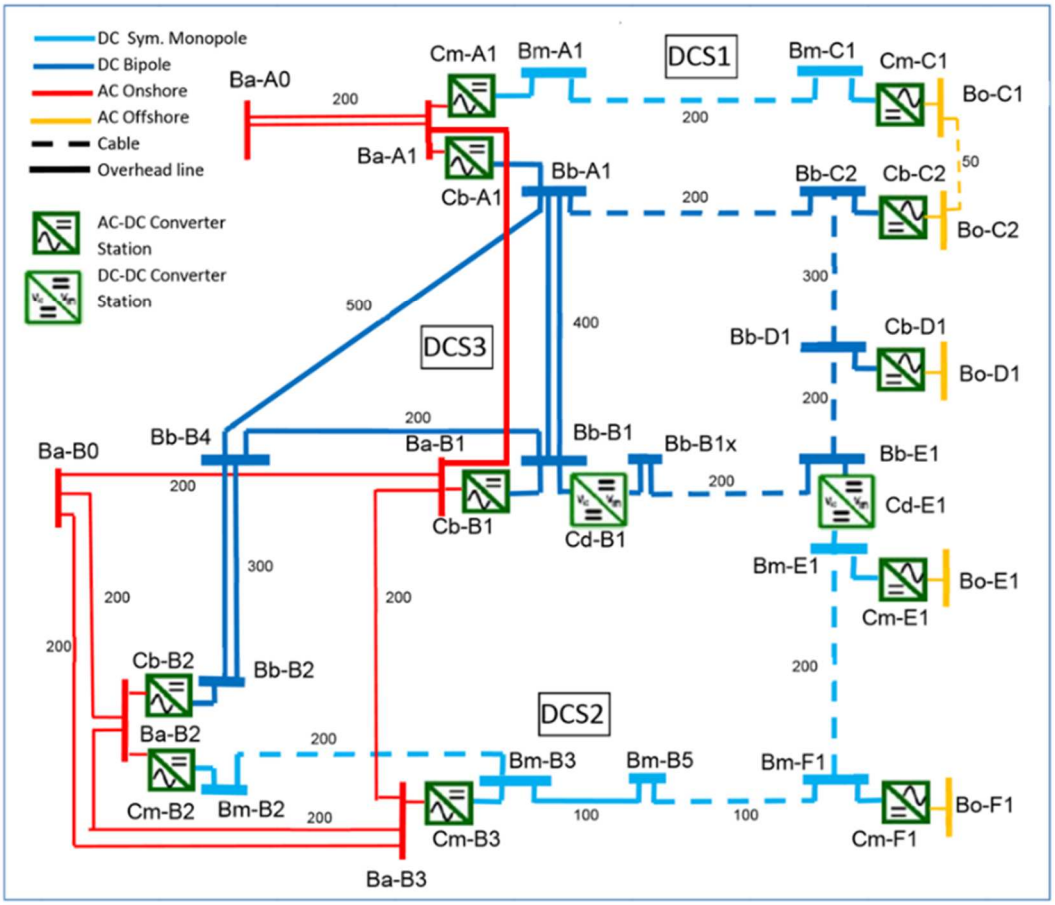


Fig. 2. Hybrid HVAC/HVDC network, adapted from [11]

Table I shows HVAC-HVDC network parameters: generation and consumption data, line parameters and converter data (with operational setpoints).

Table I. HVAC-HVDC network parameters [11]

AC bus data											
Bus	A0	A1	B0	B1	B2	B3	C1	C2	D1	E1	F1
Bus type	Slack	PQ	Slack	PQ	PQ	PQ	PQ	PQ	PQ	PQ	PQ
Generation [MW]	-	2000	-	1000	1000	1000	500	500	1000	0	500
Load [MW]	-	1000	-	2200	2300	1900	0	0	0	100	0

AC-DC converter data								
AC-DC conv. station	Power rating [MVA]	Converter arm reactance [mH]	Transf. leakage reactance [mH]	Transf. resistance [Ω]	Transf. primary voltage [kV]	Transf. secondary voltage [kV]	Operation mode setpoints	
A1	800	29	35	0.363	380	220	Q = 0	V_{DC} = 1 pu
C1	800	58	69	0.726	145	220	V/f control	
B2	800	29	35	0.363	380	220	Q = 0	V_{DC} = 0.99 pu
B3	1200	19	23	0.242	380	220	V_{AC} = 1 pu	P = 800 MW
E1	200	116	58	29	29	19	V/f control	
F1	800	29	35	0.363	145	220	V/f control	
A1	2*1200	19	23	0.242	380	220	V_{AC} = 1 pu	V_{DC} = 1.01 pu
B1	2*1200	19	23	0.242	380	220	V_{AC} = 1 pu	P = 1900 MW
B2	2*1200	29	35	0.363	380	220	V_{AC} = 1 pu	P = 1700 MW
C2	2*400	58	69	0.726	145	220	V_{AC} = 1 pu	P = −600 MW
D1	2*800	29	35	0.363	145	220	V/f control	

AC and DC line data					
Line data	r_0 [Ω/km]	l [mH/km]	c [μF/km]	g_0 [μS/km]	Max. current [A]
DC OHL ± 400 kV	0.0114	0.9356	0.0123	-	3500
DC OHL ± 200 kV	0.0133	0.8273	0.0139	-	3000
DC cable ± 400 kV	0.011	2.615	0.1908	0.048	2265
DC cable ± 200 kV	0.011	2.615	0.2185	0.055	1962
AC cable 145 kV	0.0843	0.2526	0.1837	0.041	715
AC OHL 380 kV	0.02	0.8532	0.0135	-	3555

Analysis of the Steady-State Results

Results from the steady-state calculation are visually presented in Figure 3, where the active power, reactive power, DC power, AC and DC voltages as well as AC voltage angle are brought forward.

In the DC region, the lines with the highest loading are *B3-F1* (691.9 MW), *D1-E1* (911.31 MW) and *B4-A1* (983.4 MW). Furthermore, all DC node voltages retain a voltage value that is close to 1 p.u., within the range of 0.983 p.u. and 1.015 p.u. Power-flow in the DC network is effectively regulated with the help of DC/DC converters, that ensure no lines are overloaded.

In the AC region, because most of the converters operate in constant voltage control mode, the AC voltages remain close to 1 p.u. Furthermore, because there is a high consumption in the north-western zone of the AC onshore grid, most of the power required in *A1* node is brought through AC lines from the slack bus, which leads to a voltage angle in node *A1* of -4.3° (which is the highest value considering the entire AC onshore grid. However, the highest loading can be observed in the south-western AC

network (approx. 3400 MW), and it is covered by slack bus *Ba-Bo*, and with contribution from north-western AC zone and the off-shore AC zone (through the DC network).

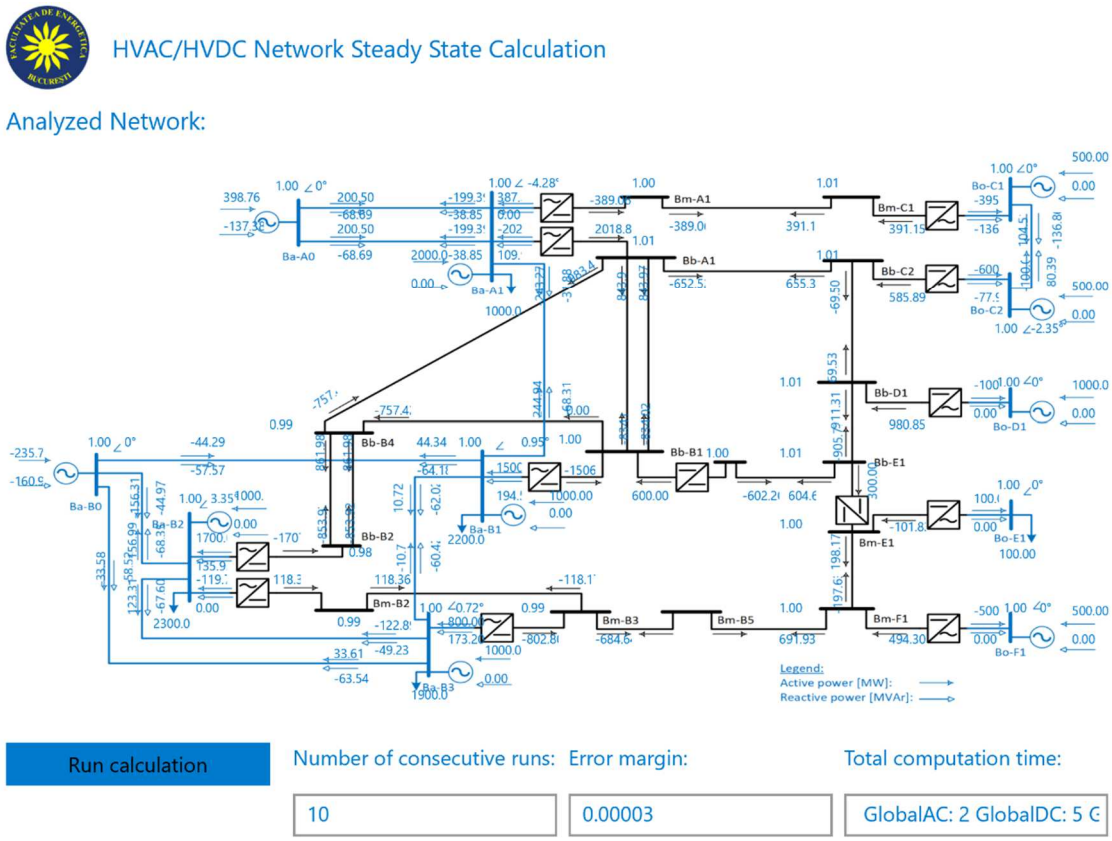


Fig. 3. Software interface with steady-state results

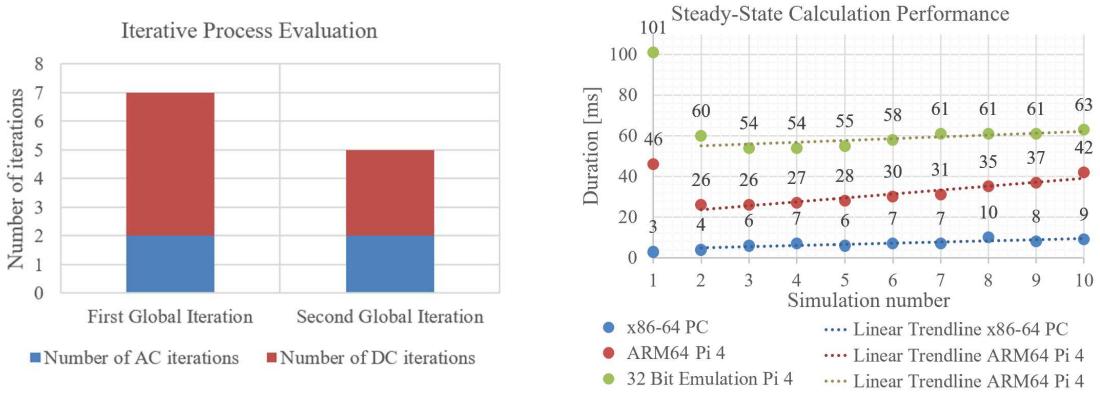
Evaluation of ARM Device Performance

The steady-state calculation, performed on both types of devices shows that for the specified error, there are a total of 2 global iterations. This demonstrates the effectiveness of the Newton-Raphson method, even applied for a hybrid HVAC/HVDC network. The number of iterations which are necessary for the AC zone and for the DC zone are summarized in Figure 4.*a*. One global iteration consists of several iterations for the AC zone and a number of iterations for the DC zone. In the second global iteration, the last values computed for the DC zone (with AC interface) are used as initialization parameters for the AC zone.

Next, an analysis is performed regarding the time necessary to perform 10 consecutive runs (it is done on both devices). The number of milliseconds for each run is shown in Figure 4.*b*. One simulation amounts to a complete run of the steady-state calculations, for both the AC and the DC zones (two global iterations). Figure 4.*b* also shows the linear trendlines for all scenarios, without including the first simulation run to eliminate possible memory allocation delays.

It can be observed that the best performing platform is the x86-64 PC, followed by the native ARM64 Raspberry Pi 4. The difference between the two solutions amounts to approximately 22 ms. Furthermore, the least performing platform is the 32bit emulation on the Raspberry Pi 4, which computes the same calculations in additional 55 ms compared to x86-64.

Another observation relates to the fact that there is a higher duration for the first run, for the native and emulation approach on Raspberry Pi 4. More specifically, the first simulation shows doubled duration for the ARM64 solution, and a 70% increase in the time necessary to run the first run on the 32bit emulation.



a) b)

Fig. 4. a) Overview of the iterative process; b) Simulation time for three use-case scenarios

Additionally, as the number of simulations increase, so does duration. This is an expected behavior, because after each run, processor temperature increases. Therefore, in order to remain within thermal limits, the computing unit decreases frequency. This is visible for all solutions.

The case-study analysis shows that there are differences regarding simulation time, between the three approaches. Furthermore, the most efficient manner in which steady-state simulations should be performed on an ARM device is to compile a program for the native environment (which is ARM64). However, time benefits for the classical x86-64 over ARM64 are less than 43 ms, which can be considered negligible. Therefore, taking into consideration that an ARM device, such as the Raspberry Pi 4, costs less than 60 USD and is able to compute complex steady-state simulations in the order of milliseconds, it is clearly a feasible option.

Even though for longer-term computations thermal limits are stricter in this form-factor (PCs have extensive active and passive cooling capabilities, while the Raspberry Pi 4 only relies on natural convection), good steady-state computation performance on a Raspberry Pi 4 shows that there is potential regarding the possibility to deploy dynamic simulations on this device,

Another relevant aspect concerns the fact that Raspberry Pi 4 provides 40 general-purpose input/output (GPIO) pins, which can be used to communicate with other equipment. As such, by combining the steady-state capabilities with the possibility to interact with other devices in a supervisory control and data acquisition (SCADA) system, it is possible to envision scenarios in which an OS based Raspberry Pi 4 can be converted into a network state estimator.

Finally, even though the proposed solution from the case study was applied on hybrid HVAC/HVDC transmission networks, it can also be extended to hybrid distribution networks.

Conclusion

The scope of this paper was to evaluate the feasibility regarding the usage of an OS based ARM device for steady-state calculations of complex HVAC/HVDC networks, considering the expansion of such devices beyond traditional IoT applications.

The first part of this article consisted of a review of methods which enable steady-state calculations for hybrid networks, consisting of multiple HVAC and HVDC zones. It was concluded that one flexible approach is the usage of a sequential algorithm, which alternates between the AC zones and the DC zones and relies on the iterative Newton-Raphson method.

In the second part, focus was directed towards the actual analysis of the computing performance of an OS based Raspberry Pi 4 for steady-state calculations. This original concept was steadily analyzed by implementing a software that was built on the C# programming language, in the Universal Windows Platform. It was demonstrated that, even though classical approaches are faster compared to the ARM solution, the differences are negligible (in the order of tens of milliseconds).

Summing up, the study performed in this article validated the feasibility of using an OS based ARM device for steady-state calculation using the sequential method and Newton-Raphson.

References

[1] Beerten J., Cole S. and Belmans R.: Generalized Steady-State VSC MTDC Model for Sequential AC/DC Power Flow Algorithms, IEEE Transactions on Power Systems, vol. 27, no 2, May 2012

[2] Damian I. C.: Supply of Large Cities Using Modular Multilevel High Voltage Direct Current Converters. PhD Thesis, University Politehnica of Bucharest, 2020

[3] Eremia M., Liu C. C. and Edris A. A. (Eds.): Advanced Solutions in Power Systems - HVDC, FACTS and Artificial Intelligence, Hoboken, New Jersey: IEEE and Wiley Publishing Press, 2016

[4] Ghael H. D., Solanki L., Sahu G.: A Review Paper on Raspberry Pi and its Applications, International Journal of Advances in Engineering and Management, Vol. 2, Issue 12, pp. 225-227, 2020

[5] Eremia M. (Ed.): Electric Power Systems: Electric Networks, Bucharest, Romanian Academy Publishing, 2006

[6] Chaudhuri N. and Chaudhuri B.: Multi-terminal Direct-Current Grids, Wiley & Sons, 2014

[7] Jovcic D.: High Voltage Direct Current Transmission - Converters, Systems and DC Grids, Second Edition, Wiley & Sons, 2019

[8] Hertem D. V., Gomis-Bellmunt O. and Liang J.: HVDC Grids for Offshore and Supergrid of the Future, Hoboken, New Jersey: John Wiley & Sons, 2016

[9] Kim C. K., Moon S. I., Hur K., Kim J. M. and Jang G.: HVDC Transmission - VSC HVDC Based MMC Topology in Power Systems, World Scientific, 2021

[10] Windows on Raspberry Project. [Online]. Available: https://www.worproject.ml/ (accessed on December 2020).

[11] Wachal R., Jindal A., Dennetiere S., Saad H., Rui O., Cole S., Barnes M., Zhang L., Song Z., Jardini J., Garcia J. C., Mosallat F., Suriyaarachich H., Le-Huy P., Totterdell A., Zeni L., Kodsi S., Deepak T., Thepparat P., Beddard T., Velasquez J., D'Arco S., Morales A., Kono Y., Vrana T. K. and Yanh Y.: Guide for the Development of Models for HVDC Converters in a HVDC Grid, Working Group B4.57, Cigre, December 2014