EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Actor based Diakoptics (A-Diakoptics) suite for OpenDSS

Davis Montenegro
Roger Dugan

Last update 02-28-2019

Diakoptics based on Actors (A-Diakoptics) combines two computing techniques from different engineering fields: Diakoptics and the actor model. Diakoptics is a mathematical method for tearing networks. The actor model is used to coordinate the interaction between sub-circuits.

A-Diakoptics is a technique that seeks to simplify the power flow problem to achieve a faster solution at each simulation step. Consequently, the total time reduction when performing QSTS will be evident at each simulation step with A-Diakoptics.

This technique has been implemented in OpenDSS in version 8.5 using a simplification that will be explained later in this document. The suite integrated in version 8.5 includes automated circuit tearing and other tools to execute and debug the method step by step. Since the simplification implemented in OpenDSS is relatively new, we will continue to improve the suite in time.

The simplified A-Diakoptics solution method

Initially proposed by Gabriel Kron and later used and modified by other authors [1, 2]. Diakoptics is a technique for tearing large physical circuits into a number of sub-circuits to reduce the modeling complexity and accelerate the solution of the power flow problem using a computer network. Each computer will handle a separate piece of the circuit to find a total solution. Using modern multi-core computers this technique can be used for accelerating QSTS simulations, in OpenDSS, by using the actor model as a framework for coordinating the interactions between the distributed pieces proposed in Diakoptics [3, 4].

A-Diakoptics uses the parallel processing suite inside OpenDSS to allocate and solve the separate pieces of the interconnected circuit. A-Diakoptics was proposed initially in 2015 [5, 6] and utilizes the power flow solution method proposed in OpenDSS for the analysis.

The power flow problem in OpenDSS

While the power flow problem is probably the most common problem solved with the program, the OpenDSS is not best characterized as a power flow program. Its heritage is from general-purpose power system harmonics analysis tools. Thus, it works differently than most existing power flow tools. This heritage also gives it some unique and powerful capabilities for modeling complex electrical circuits. The program was originally designed to perform nearly all aspects of distribution planning for distributed generation (DG), which includes harmonics analysis. It is relatively easy to make a harmonics analysis program solve a power flow, while it can be quite difficult to make a power flow program perform harmonics analysis. To learn more about how the algorithm works for the power flow problem, see “Putting It All Together” below. [Where is this?]

The OpenDSS program is designed to perform a basic distribution-style power flow in which the bulk power system is the dominant source of energy. However, it differs from the traditional radial circuit solvers in that it solves networked (meshed) distribution systems as easily as radial systems. It is intended to be used for distribution companies that may also have transmission or subtransmission systems. Therefore, it can also be used to solve small- to medium-sized networks with a transmission-style power flow.

Nearly all variables in the formulation result in a matrix or an array (vector) to represent a multiphase system. Many of the variables are complex numbers representing the common phasor notation used in frequency-domain ac power system analysis.

OpenDSS uses a fairly standard Nodal Admittance formulation that can be found documented in many basic power system analysis texts. The Arrillaga and Watson textbook is useful for understanding this because it also develops the admittance models for harmonics analysis similarly to how OpenDSS is formulated.

A primitive admittance matrix, *Yprim*, is computed for each circuit element in the model. These small matrices are used to construct the main system admittance matrix, *Ysystem*, that knits the circuit model together. The solution is mainly focused on solving the nonlinear system admittance equation of the form:

$$I_{PC}(E) = Y_{system}E$$

where,

$$I_{PC}(E) = Compensation\ currents\ from\ Power\ Conversion\ (PC) elements\ in\ the\ circuit$$

The currents injected into the circuit from the *PC* elements, $I_{PC}(E)$, are a function of voltage as indicated and represent the nonlinear portion of the currents from elements such as Load, Generator, PVsystem, and Storage.

There are a number of ways this set of nonlinear equations could be solved. The most popular way in OpenDSS is a simple fixed point method that can be written concisely [7]:

$$E_{n+1} = \left[Y_{system}\right]^{-1} I_{PC}(E_n) \quad n = 0,1,2 \ldots until\ converged$$

From here it can be inferred that every time an expression like the following is found:

$$E = \left[Y_{system}\right]^{-1} I$$

there is an OpenDSS solver, called an *actor* in OpenDSS' parallel processing suite, and this is the basis for the A-Diakoptics analysis.

Simplified A-Diakoptics

An initial form of A-Diakoptics was developed in [5, 6]. From there, the general expression for describing the interactions between the actors (sub-circuits) and the coordinator is as follows:

$$E_T = Z_{TT} I_{0(n-1)} - Z_{TC} Z_{CC}^{-1} Z_{CT} I_{0(n)}$$

E_T is the total solution of the system (the voltages in all the nodes of the system). I_0 is the vector containing the currents injected by the PC elements, and the time instants *n* and *n +1* are discrete

time instants to describe the different times in which the vector of currents is calculated [6]. Z_{TT} is the trees matrix and contains the admittance matrixes for all the sub-circuits that contained in the interconnected system after the partitioning. The form of Z_{TT} is as follows:

$$Z_{TT} = \begin{bmatrix} [Y_1]^{-1} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & [Y_n]^{-1} \end{bmatrix} n = 0,1,2 \ldots \#\ of\ sub-circuits$$

The sub-circuits contained in Z_{TT} are not interconnected in any way. The connections are through external interfaces that define the relationship between them. Z_{TC}, Z_{CT} and Z_{CC} are interfacing matrixes for interfacing the separate subsystems using a graph defined by the contours matrix (*C*) as described in [6, 8]. $Z_{TT}I_0(n-1)$ corresponds to the solutions delivered when solving the sub-systems and $Z_{TC}Z_{CC}^{-1}Z_{CT}I_0(n)$ is the interconnection matrix to find the total solution to the power flow problem.

As can be seen at this point, the form of Z_{TT} proposes that multiple OpenDSS solvers can find independent partial solutions that when complemented with another set of matrixes, can calculate the voltages across the interconnected power system. However, in this approach the interconnection matrix is a dense matrix that operates on the injection currents vector provided by the latest solution of the sub-systems.

Nowadays, sparse matrix solvers are very efficient and to operate with a dense matrix is not desirable. The aim in this part of the project is to simplify the expression that defines the interconnection matrix to find a sparse equivalent that can be solved using the KLUSolve module already employed in OpenDSS.

Assuming that the times in which the sub-systems are solved and the interconnection matrix is operated are the same and fit into the same time window (ideally), A-Diakoptics can be reformulated as:

$$E_T = Z_{TT}I_{0(n)} - Z_{TC}Z_{CC}^{-1}Z_{CT}I_{0(n)} \qquad (*)$$

Additionally, in OpenDSS the interconnected feeder is solved using:

$$E_T = [Y_{II}]^{-1}I_0$$

Where Y_{II} is the Y_{Bus} matrix that describes the interconnected feeder. Equating the two previous equations the expression results in:

$$[Y_{II}]^{-1}I_{0(n)} = Z_{TT}I_{0(n)} - Z_{TC}Z_{CC}^{-1}Z_{CT}I_{0(n)}$$

This new expression can be taken to the admittance domain since Z_{TT} is built using the Y_{Bus} matrixes that describe each one of the sub-systems created after tearing the interconnected feeder [5, 6]. The new equation is reformulated as:

$$[Y_{II}]^{-1}I_{0(n)} = [Y_{TT}]^{-1}I_{0(n)} - Z_{TC}Z_{CC}^{-1}Z_{CT}I_{0(n)}$$

$$[Y_{II}]^{-1} = [Y_{TT}]^{-1} - Z_{TC}Z_{CC}^{-1}Z_{CT}$$

Then the interconnection matrix can be reformulated as:

$$Z_{TC}Z_{CC}^{-1}Z_{CT} = [Y_{XX}]^{-1} = [Y_{TT}]^{-1} - [Y_{II}]^{-1}$$

This new formulation proposes that the interconnection matrix is equal to an augmented representation of the link branches between the sub-systems. To support this conclusion take the simplification proposed by Happ in [8] where:

$$e_{c\prime} = -Z_{CT} I_{0(n)}$$

$$Z_{CT} = C^T Z_{TT} \rightarrow e_{c\prime} = -C^T Z_{TT} I_{0(n)}$$

This expression is similar to the partial solution formulation of the Diakoptics equation but including the contours matrix, then if the partial solution is called $E_{T(0)}$ it is possible to say:

$$e_{c\prime} = -C^T E_{T(0)}$$

On the other hand, Z_{TC}, which is the non-conjugate transposed of Z_{CT}, is calculated as follows:

$$Z_{TC} = Z_{TT} C$$

Replacing the new equivalences for Z_{CT} and Z_{TC} the equation proposed in (1) is reformulated as follows:

$$E_T = Z_{TT} I_{0(n)} + Z_{TT} I_c$$

Or in terms of sparse matrices:

$$E_T = [Y_{TT}]^{-1} I_{0(n)} + [Y_{TT}]^{-1} I_c \qquad (**)$$

Where:

$$I_c = -C Z_{CC}^{-1} C^T E_{T(0)}$$

The new equation described in (2) establishes that the same actor can calculate its partial and complementary solutions using information calculated using the interconnection matrix Z_{CC}. Z_{CC} is a small matrix that contains the information about the link branches and the calculation of I_c does not represent a significant computational burden in medium and large-scale circuits. With this approach, all the matrix calculations are made using sparse matrix solvers, reducing the computational burden.

Using the A-Diakoptics suite in OpenDSS

The A-Diakoptics suite is an instruction set for driving simulations using A-Diakoptics in OpenDSS. This instruction set is available from version 8.5 and later. The instructions are as follows:

Command	Description
Set ADiakoptics = XX	Activates/deactivates the A-Diakoptics suite. XX can be False/True o Yes/No. To activate A-Diakoptics there must be a system loaded into memory in actor 1 (default) and the circuit needs to be solved once in snapshot mode. When enabled, A-Diakoptics will partition the system using the number of sub-circuits specified with *set num_subcircuits=$$*, where *$$* is the number of sub-circuits. Then, the sub-circuits will be compiled and loaded into memory for the simulation. See "Circuit partitioning" for details on the output. When deactivated (*set ADiakoptics=No*), no action is taken and the crated actors will remain in memory.
Get ADiakoptics	Returns Yes/No to indicate if the A-Diakoptics suite is active
Get LinkBranches	After activating A-Diakoptics, the user can use this instruction to get the names of the link branches used for partitioning the circuit. It is required that the active actor is 1 to obtain the correct values. The partition is made using MeTIS [9] V4.

Export Contours	Exports the contours matrix calculated in compressed coordinated format. A-Diakoptics needs to be active.
Export ZLL	Exports the Link branches matrix calculated in compressed coordinated. A-Diakoptics needs to be active. See [5, 10] for details.
Export ZCC	Exports the connections matrix calculated in compressed coordinated. A-Diakoptics needs to be active. See [5, 10] for details.
Export Y4	Exports the inverse of ZCC (Y4 = Z_{CC}^{-1}) in compressed coordinated format. A-Diakoptics needs to be active.

Circuit partitioning

After activating A-Diakoptics OpenDSS will deliver an output describing the result of the initialization. Consider the following lines of code for enabling A-Diakoptics using 4 sub-circuits (the circuit has been compiled and solved in snapshot mode previously):

```
set Num_Subcircuits=4
set ADiakoptics=True
```

If the partitioning and initialization of the circuit was correct, the output from OpenDSS will be as shown in Figure 1.

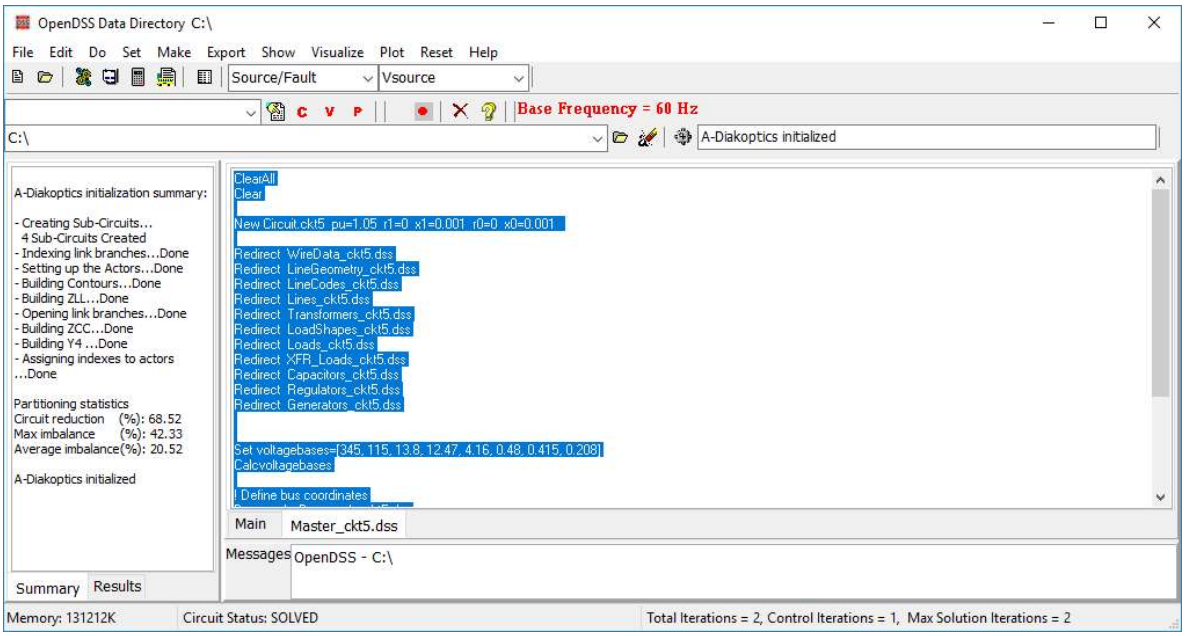

Figure 1. Circuit partitioning successful

As can be seen in Figure 1, the summary window will display all the steps carried out to initialize A-Diakoptics including the partitioning statistics (see *partitioning statistics*). At this point the system is ready to solve in A-Diakoptics mode. The following considerations need to be taken:

1. No monitors or energy meters can be created in actor 1. Actor 1 is the simulation coordinator and it hosts Y4 and the vectors for calculating *Ic*. Actor 1 also hosts the admittance matrix of the interconnected circuit. The total voltages are transmitted to actor 1 from other actors when the simulation step is completed, however, control actions are performed on the sub-circuit actors, making difficult to monitor those actions from actor 1. This is a feature on which development is continuing. Nevertheless, voltages, powers and currents can be extracted from Actor 1 to visualize the variations on the system. The control actions can be monitored using the sub-circuit actors.

2. Actor 1 is the simulation coordinator; the others are slaves of the coordinator.
3. The number of actors depends on the number of sub circuits configured by the user. If the number of sub-circuits set by the user overpasses the number of CPUs – 2 in the local PC, the Initialization algorithm will force the number of sub-circuits to the number of CPUs – 2.
4. The CPUs are assigned automatically. If the user wants a better performance (e.g. 1 thread per Core) it is necessary to do the redistribution manually.
5. The algorithm is only implemented for the standalone OpenDSS.EXE version; once there is a stable version it will be implemented in the other interfaces.
6. DO NOT use the wait command when using A-Diakoptics, the simulation coordinator will use it internally with some variations. Instead, use the *get ActorProgress* command to check the simulation progress.

Error messages

The A-Diakoptics initialization is a complex process and has multiple moving pieces that may cause an error in the initialization, making the initialization unsuccessful and aborting A-Diakoptics. Sometimes when creating the sub-circuits, one or more circuits cannot be solved, either because the circuit is invalid or something is missing. In this case the message at the summary window in OpenDSS is as follows:

```
A-Diakoptics initialization summary:

- Creating Sub-Circuits...
  6 Sub-Circuits Created
- Indexing link branches...Done
- Setting up the Actors...Error
One or more sub-systems cannot be compiled

One or more errors found
```

In this case the A-Diakoptics initialization failed and as a result, A-Diakoptics was not enabled.

OpenDSS uses MeTIS for the circuit partitioning. Sometimes MeTIS suggests a PD element different from a Line as a link branch, which is an error. In that case the output at the summary window will be:

```
A-Diakoptics initialization summary:

- Creating Sub-Circuits...
  6 Sub-Circuits Created
- Indexing link branches...Done
- Setting up the Actors...Done
- Building Contours...Error
One or more link branches are not lines

One or more errors found
```

Again, the A-Diakoptics initialization failed and A-Diakoptics was not enabled. In this case the best solution is to propose a different number of sub-circuits until a valid partitioning is achieved. The partitioning can be validated using the partitioning statistics information.

Partitioning statistics

After a successful circuit partitioning OpenDSS will calculate some statistics to help the user to understand and estimate what to expect from the circuit tearing. There are 3 indicators delivered by OpenDSS called the circuit reduction, the maximum imbalance and the average imbalance.

The circuit reduction is the relationship between the number of nodes in the interconnected circuit and the largest sub-circuit created after partitioning the circuit. The circuit reduction is calculated as follows:

$$Circuit\ reduction\ (\%) = \left(1 - \frac{\max(actor_m.NumNodes|_{m=2}^{N})}{Total\_NumNodes}\right)100 \qquad N = \#\ sub-circuits$$

It is the largest sub-circuit that will take longer to solve (normally), determining the total time required for solving a simulation step.

The maximum imbalance and the average imbalance are 2 metrics oriented to illustrate the level of balance between sub-circuits. The maximum imbalance is calculated considering the relationship between the sizes of all the sub-circuits and the largest sub-circuit after the partitioning.

$$Max\ imbalance\ (\%) = \left(1 - max\left(\frac{actor_m.NumNodes}{Largest.NumNodes}\Big|_{m=2}^{N}\right)\right)100 \qquad N = \#\ sub-circuits$$

The average imbalance is the average of the all the imbalances considering the largest sub-circuit as reference.

$$avg\ imbalance\ (\%) = mean\left(\left[1 - \frac{actor_m.NumNodes}{Largest.NumNodes}\right]\Big|_{m=2}^{N}\right)100 \qquad N = \#\ sub-circuits$$

The aim with these metrics is to keep them as low as possible after partitioning the circuit. Ideally, the average and maximum imbalances should be 0. However, that may happen only in very special situations. If the maximum imbalance is lower than 50 % it means that the tearing is acceptable. For example, consider the following statistics after tearing the circuit using 4 sub-circuits:

```
Circuit reduction    (%): 68.52
Max imbalance        (%): 42.33
Average imbalance    (%): 20.52
```

These statistics reflect that the circuit had a reduction greater than the 50% and the maximum imbalance is not perfect but is below 50%. In another scenario using only 2 sub-circuits, the output is as follows:

```
Circuit reduction    (%): 46.34
Max imbalance        (%): 13.64
Average imbalance    (%): 6.818
```

The imbalance ratio is much better than when using 4 sub-circuits, however, the reduction ratio is lower. Finally, consider a partitioning using 5 sub-circuits in the same system, the output is as follows:

```
Circuit reduction    (%): 60.95
Max imbalance        (%): 94.04
Average imbalance    (%): 48.67
```

In this case, the circuit reduction is not higher than using 4 sub-circuits and the maximum and average imbalances growth significantly, indicating that the circuit partitioning is not better using more partitions. To check the number of buses and nodes after tearing the circuit using the A-

Diakoptics initialization execute a *SolveAll* command, it will solve each sub-circuit in parallel showing each circuit's features in the summary window as shown in.

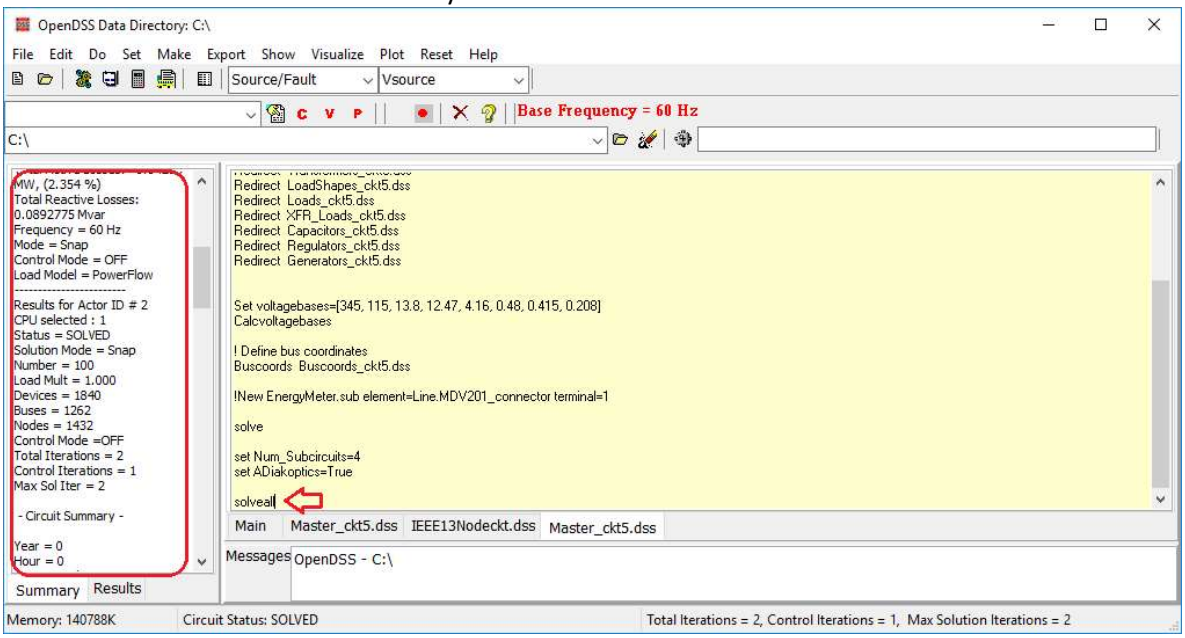

Figure 2. Checking the sub-circuits' features

Actors distribution in memory

After a successful initialization of A-Diakoptics, the total number of actors created will be the number of sub-circuits + 1. Actor 1 is the simulation coordinator actor and contains all the matrices needed for A-Diakoptics: C, Z_{LL}, Z_{CC} and Y_4. Actor 1 also contains the total solution vector after each simulation step. The other actors are the sub-circuits that compose the A-Diakoptics environment. Each compiled sub-circuit will reside in a subfolder inside the original circuit model folder in the hard drive, OpenDSS will create a subfolder called *Torn_Circuit* that contains the models for each zone and the interconnected circuit separately.

The algorithm is executed in OpenDSS as shown in Figure 3. There is a condition that needs to be met when the actors return the partial solution to the coordinator. A-Diakoptics respects the existence of power injection elements across the circuit and since after partitioning the circuit these sources are located in different sub-circuits. It is expected that these active elements (in OpenDSS a VSource or ISource) are capable of exciting their sub-circuits. However, not all the sub-circuits have an active element within (in OpenDSS PVSystems, Storage and other DER are represented as dynamic negative loads for QSTS), requiring an artificial excitation for solving the sub-circuit and to consider its load variability, which will be transmitted to the simulation coordinator.

But the inclusion of the artificial exciter is a feature that cannot be transmitted to the coordinator since it supposes that there is no excitation in the sub-circuit, and A-Diakoptics is sensitive to it. This issue is solved using the following condition:

$$E_{(0)m} = \begin{cases} E_{(0)m(n)} \; if \; the \; actor \; contains \; AE \\ E_{(0)m(n)} - E_{Tm(n-1)} \; if \; the \; actor \; does \; not \; contain \; AE \end{cases}$$

AE = Active Element

Where $E_{(0)m}$ is the partial solution for the actor m, $E_{(0)m(n)}$ is the partial solution found by the actor m for the simulation step n, and $E_{Tm(n-1)}$ is the total solution found by the actor m in the previous

simulation step. Using this relationship, the variations in voltage introduced by loads variation and DER can be transmitted to the coordinator for the next simulation step.

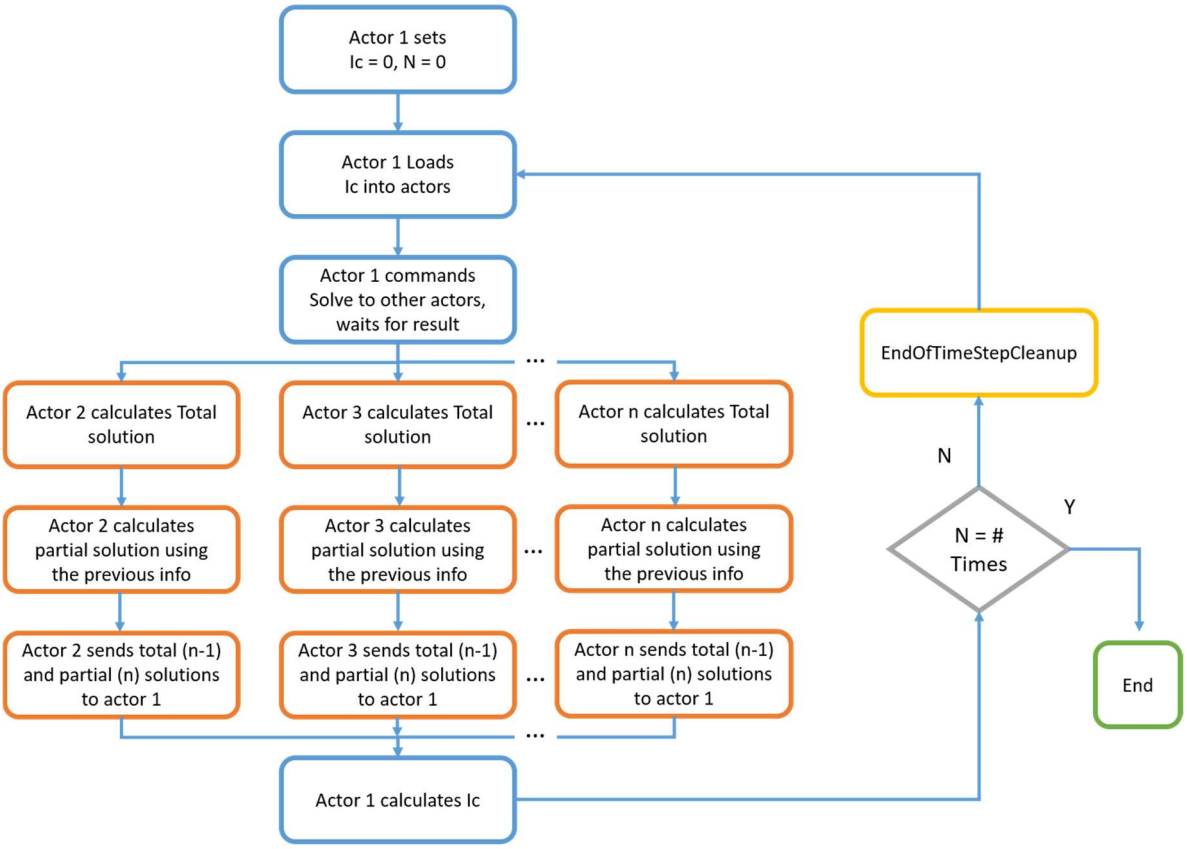

Figure 3. A-Diakoptics algorithm in OpenDSS

Solving the Circuit using A-Diakoptics

To solve the circuit, make sure that the active actor is Actor 1, then set the simulation mode and the simulation features. Finally, execute the *solve* command.

```
set activeactor=1
set mode=daily stepsize=1m number=1440
solve
```

It is possible to navigate between actors using the *set activeactor=$$* command to change actor features such as CPU or to add new elements such as monitors and energy meters (These features will be transported into Actor 1 in future implementations).

A-Diakoptics performance

Two circuits have been selected to evaluate the performance of this technique. The circuits selected include both small and large scale circuits: IEEE 123 node test system and IEEE 8500 node system.

A-Diakoptics on small scale circuits

The IEEE 123 node system is partitioned in 2 and 4 actors and then solved using the A-Diakoptics suite in OpenDSS. After the partitioning, the circuit is solved using a default yearly simulation (8760, 1 hour simulation step interval) to evaluate the performance in terms of the computational time required to complete the QSTS simulation. The result are as follows:

Table 1. Data obtained for small scale circuit

Partition	*Total time (µs)*	*Ckt. Reduction (%)*	*Avg. Imbalance (%)*	*Max. Imbalance (%)*
4 actors	1097360	53.6	46.12	90.7
2 actors	1100754	49.28	2.837	2.837
1 actor	1874862	0	0	0

As can be seen in Table 1, the circuit reduction is not significantly improved when increasing the number of partitions; however, the imbalance between partitions gets significantly affected. The growth of the imbalance means that the partitioning process was not able to tear the circuit properly and some of the partitions are significantly smaller than others, making the performance of the simulation to be proportional to the size of the largest partition as shown in Figure 4.

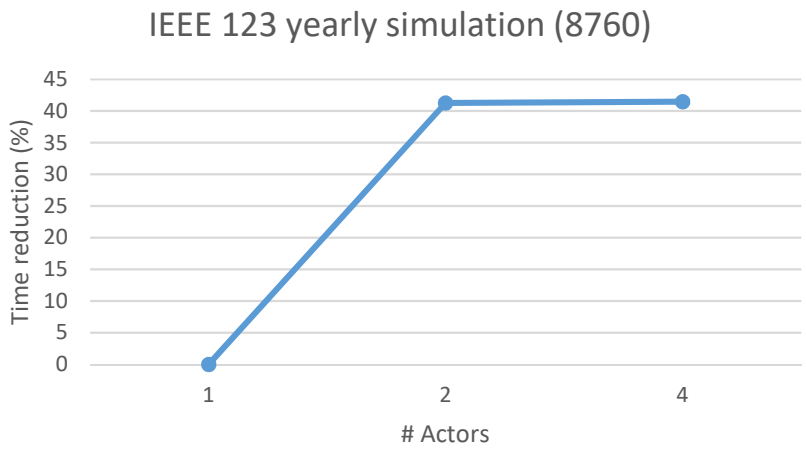

Figure 4. Time reduction achieved for the IEEE 123 node test system

As can be seen in Figure 4, there is a significant solution time improvement by tearing the circuit into 2 partitions. Nevertheless, the simulation time reduction does not get an important impact when doubling the number of partitions, which is directly related to the partition statistics as shown in Figure 5.

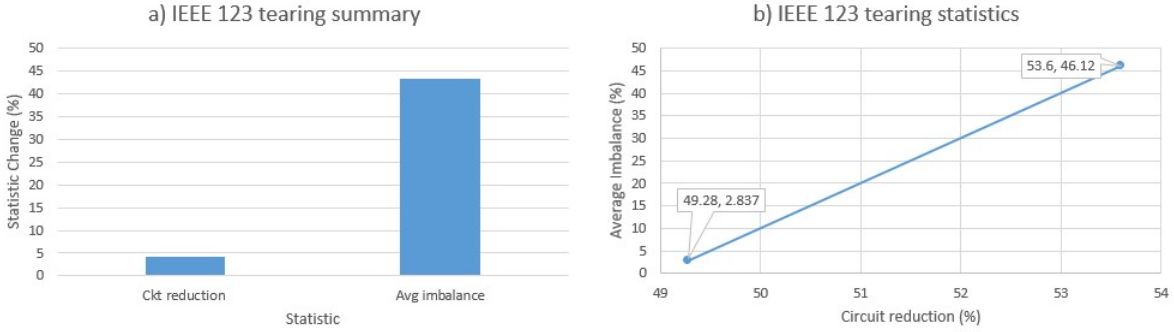

Figure 5. Tearing statistics for IEEE 123 node test system

Figure 5 displays how the tearing statistics affect directly the simulation's performance, in the first place, the circuit reduction does not show an important variation between 2 and 4 actors while significantly increasing the partition imbalance. The variation on the circuit reduction (the largest

partition after portioning the interconnected circuit) finds a direct connection to the simulation performance according to the data presented here.

A-Diakoptics on large scale circuits

The IEEE 8500-Node test system is partitioned into 2 and 4 actors as in the previous case, the data and statistics obtained for this case are as follows:

Table 2. Data obtained when partitioning a large scale circuit

Partition	*Total time (µs)*	*Ckt. Reduction (%)*	*Avg. Imbalance (%)*	*Max. Imbalance (%)*
4 actors	14242591	69.59	45.19	76.72
2 actors	42013831	46.81	12.01	12.01
1 actor	60679939	0	0	0

As can be seen in Table 2, the circuit reduction rate increases when partitioning the circuit in more pieces, the imbalance between actors has also increased but the average and maximum values are closer than in the IEEE 123 case. As a consequence, the simulation time gets reduced substantially as shown in Figure 6.

As can be seen in Figure 6, The time reduction is continuous when increasing the number of partitions. The statistics shown in Figure 7 reveal that the improvement on the circuit size reduction and the stability on the imbalance between actors helps to improve the performance of the A-Diakoptics algorithm when dealing with large scale circuits.

The indicators proposed to evaluate the performance of the partitioning process are a good guide on the performance of the automated tearing algorithm and as a consequence, A-Diakoptics. There are more options for configuring MeTIS that need to be explored to improve the partitioning stage in order to guarantee the best performance of A-Diakoptics.

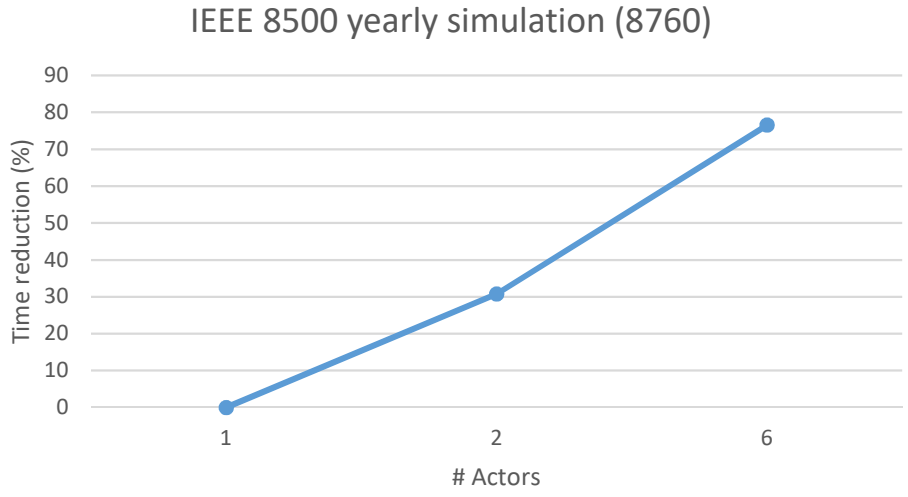

Figure 6. Time reduction achieved for the IEEE 8500 node test system

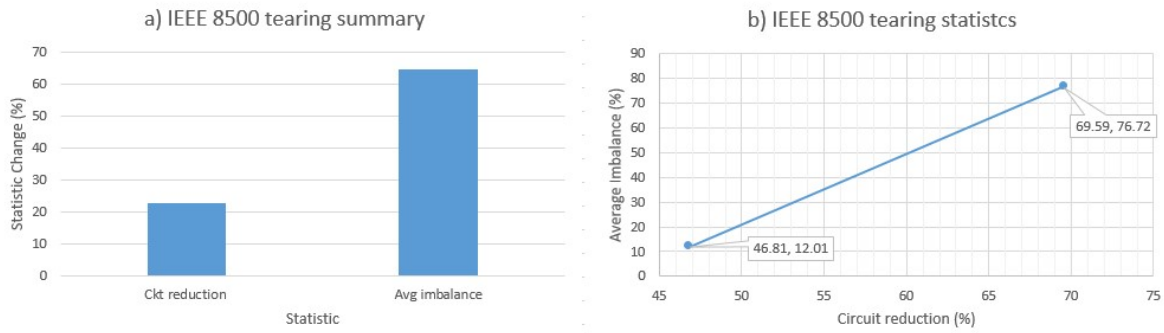

Figure 7. Tearing statistics for IEEE 8500 node test system

An important feature of the simplified A-Diakoptics algorithm proposed here is that even small scale circuits will obtain simulation time reduction, this technique based on sparse matrix operations allows to obtain improvements in terms of computational performance.

References

[1] G. Kron, "Detailed Example of Interconnecting Piece-Wise Solutions," *Journal of the Franklin Institute,* vol. 1, p. 26, 1955.

[2] G. Kron, *Diakoptics: the piecewise solution of large-scale systems*: Macdonald, 1963.

[3] C. Hewitt, "Actor Model of Computation: Scalable Robust Information Systems," in *Inconsistency Robustness 2011, Stanford University*, 2012, p. 32.

[4] C. Hewitt, E. Meijer, and C. Szyperski. (2012, 05-15). *The Actor Model (everything you wanted to know, but were afraid to ask)*. Available: http://channel9.msdn.com/Shows/Going+Deep/Hewitt-Meijer-and-Szyperski-The-Actor-Model-everything-you-wanted-to-know-but-were-afraid-to-ask

[5] D. Montenegro, G. A. Ramos, and S. Bacha, "Multilevel A-Diakoptics for the Dynamic Power-Flow Simulation of Hybrid Power Distribution Systems," *IEEE Transactions on Industrial Informatics,* vol. 12, pp. 267-276, 2016.

[6] D. Montenegro, G. A. Ramos, and S. Bacha, "A-Diakoptics for the Multicore Sequential-Time Simulation of Microgrids Within Large Distribution Systems," *IEEE Transactions on Smart Grid,* vol. 8, pp. 1211-1219, 2017.

[7] R. Dugan. (2016, OpenDSS Circuit Solution Technique. 1. Available: https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Version7/Doc/OpenDSS%20Solution%20Technique.docx

[8] H. H. Happ, *Piecewise Methods and Applications to Power Systems*: Wiley, 1980.

[9] H. Meyerhenke, P. Sanders, and C. Schulz, "Parallel Graph Partitioning for Complex Networks," in *2015 IEEE International Parallel and Distributed Processing Symposium*, 2015, pp. 1055-1064.

[10] H. H. Happ, "Z Diakoptics - Torn Subdivisions Radially Attached," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-86, pp. 751-769, 1967.