An Alternate PowerFactory Matlab Coupling Approach

Aadil Latif, Mohsin Shahzad, Peter Palensky
Energy Department
AIT Austrian Institute of Technology
Vienna, Austria
aadil.latif.fl@ait.ac.at

Wolfgang Gawlik
Institute of Energy Systems and Electrical Drives
Vienna University of Technology
Vienna, Austria
wolfgang.gawlik@tuwien.ac.at

Abstract— **PowerFactory is a powerful power system analysis tool used for simulating the electrical grid. 'Smart Grid' envisions a modernized electrical grid that uses communication to gather and act on information. The ever increasing communication and controls in power systems increases the complexity of the system. Co-simulation becomes essential to couple system simulators from different domains. This paper gives an overview of possible PowerFactory / Matlab coupling approaches. It further discusses the advantages and limitations of each of these. Additionally, an alternate coupling approach is suggested, its pros and cons are discussed. Two test cases have been implemented that highlight different advantages of the proposed method.**

Keywords—PowerFactory; Matlab; Co-Simulation; Software coupling

I. Introduction

With increasing share of renewable energy in the electricity market, there is a growing need for a power grid simulator that can cater for interdependencies between the power grid and other domains such as the communication network, weather forecasting tools etc. [1]. Many commercial and open source tools are available that are domain specific e.g. ETAP and PowerFactory for power systems simulation, OMNET++ and Opnet for communication network simulation and F5 for weather forecasting. One solution to simulate these interdependencies is coupling these domain specific tools together.

PowerFactory is a tool for analysis of the electrical power system designed for domain experts. The software package is vertically integrated catering to both transmission and distribution networks. It provides capabilities for load flow, short circuit analysis, steady state, transient , optimal power flow as well as several other studies for balanced and unbalanced systems. In PowerFactory system dynamics can be investigated using load flow, root mean square (RMS) simulation or electro-magnetic transient simulation depending on the time constant of the phenomena. DIgSILENT Programming Language (DPL) is a scripting language in PowerFactory that can be used to access objects and automate tasks e.g. short circuit sweep. DPL however cannot be used while RMS simulation is running. Custom controllers and algorithms can be implemented in PowerFactory using DIgSILENT Simulation Language (DSL) [2].

Matlab is a high level language with a large number of interdisciplinary tool boxes e.g. optimization, signal processing, statistics, image processing etc. [3]. Using co-simulation framework these toolboxes can be used in conjunction with PowerFactory to extend its capabilities. A key feature of Matlab is Simulink, a multi-domain GUI based simulation environment, which comes with a huge library for modeling both continuous and discrete time systems. This makes it an ideal tool to implement custom system models such as generators, relays etc.

Andrei Stativa et al. [4] in their work optimized the tuning parameters of power system stabilizers and its placement by implementing the controllers in PowerFactory and a multi-objective optimization algorithm in Matlab. Statistics toolbox can potentially be very useful to automate statistical analysis for results obtained from a large number of simulation runs. Matthias Stifter et al. [5] in their Co-Simulation Training Platform for Smart Grids implemented the power system in PowerFactory, components like generators and batteries in Matlab and the control system in 4DIAC.

The paper is organized as follows: Section II gives an over view of various PowerFactory/Matlab coupling schemes. Additionally, a pros and cons of each of these methods are discussed. Section III describes the coupling architecture of the proposed method. Two test cases have been implemented and discussed in Section IV to highlight different advantages of the proposed method. In Section V a conclusion is derived based on the results detailed in Section IV.

II. Coupling Schemes

PowerFactoy supports a number of external interfaces that can be used for data exchange and Matlab coupling. User requirements and ease of implementation are the main driving factors in selection of the interface used for coupling.

A. PowerFactory's builtin Matlab interface

1) PowerFactory provides a direct interface to Matlab by connecting a DSL model to a Matlab script. In this co-simulation framework Matlab runs in engine mode [2] and the simulation runs in a sequential manner. Matlab is invoked at every time step and all the parameters are sent. It then returns a time vector, state variable matrix and controller output matrix. PowerFactory uses the matrices and the time vector to calculate the derivatives and the output of the controller. Figure 1 illustrates the sequential co-simulation procedure between PowerFactory and Matlab.

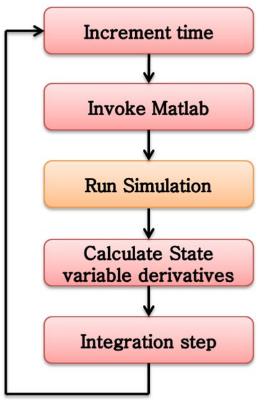


Fig. 1. Sequential co-simulation of PowerFactory and Matlab

B. *Peer-to-peer (P2P)*

1) Andrei Stativa et al. [4] used a file sharing approach to couple Matlab and PowerFactory for an optimization problem. The power system was modeled in PowerFactory and the optimization algorithm was implemented in Matlab. Figure 2 is the graphical illustraton of the coupling scheme of the two simulators. Three CSV files, one dedicated for transfering data from Matlab to PowerFactory, one to transfer results from PowerFactory to Matlab and one to switch between the two simulators, were used in this coupling scheme.

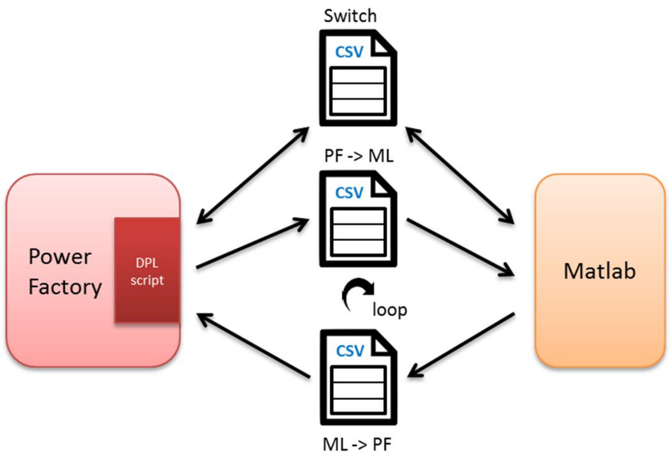


Fig. 2. File sharing between Matlab and PowerFactory

2) PowerFactory provides OPC interface for asynchronous data exchange. It is popular interface for industrial networks for SCADA and control system implementation. OPC interfacing requires a dedicated OPC server and PowerFactory to run as an OPC client. The frequency of data transfer is user defined. Matlab has an OPC toolbox that enables it to connect to an OPC server. Figure 3 illustrate a possible PowerFactory/Matlab coupling scheme.

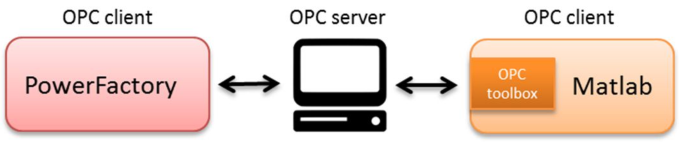


Fig. 3. Matlab and PowerFactory communicate as OPC client

3) A powerful feature of PowerFactory is the ability of a user to include external C++ functions in both DSL blocks and DPL scripts. [5] implemented a socket comunication example using an external DLL to open a socket to a server and transmit measured data. Matlab provides builtin commands for socket communication. The same principle can be extended to implement a bidirectional communication with Matlab via sockets. Figure 4 shows a scheme for coupling PowerFactory DSL block and Matlab using sockets.

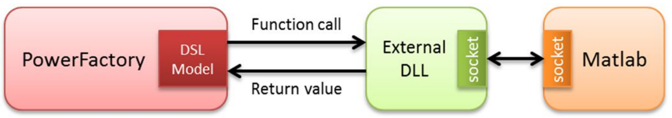


Fig. 4. PowerFactory and Matlab coupling via sockets using external DLL

C. *Coupling PowerFactory and Matlab using Libray (dll)*

1) PowerFactory's application programming interface (API) is perhaps the most power interfacing scheme. The API provides 3rd party applications access to PowerFactory data models, internal calulations and simulation functions. The API runs PowerFactory in engine mode. The C++ application can then communicate with Matlab using several different techniques as shown in Figure 5.

a) Matlab provides an API for C++ that can be used to couple the two simulators. In this coupling scheme both PowerFactory and Matlab are accessed via API and the C++ application instantiates both the simulators.

b) The C++ application can use a socket to communicate with Matlab, making the C++ controlling application and PowerFactory once again will be instantiated using the API.

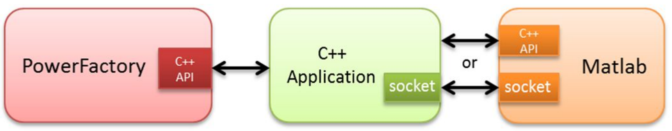


Fig. 5. PowerFactory and Matlab coupling using the API

III. PF-ML-Coupler

To be able to create an instance of PowerFactory from within Matlab a wrapper "PF-ML-Coupler" has been written and assembled as a .NET object. The wrapper uses the API interface and extends its low level functionality. These functionalities include:

- Running balanced and unbalanced load flows
- Control over RMS and transient simulation
- Read / write object values
- Read PowerFactory internal calculations
- Running other power system analysis tools provided by PowerFactory i.e. modal analysis, reliability analysis, optimal power flow etc.
- Exporting result files
- Create / delete objects within PowerFactory.

Matlab version R2009a and later can dynamically load DLLs assemblies from .NET Framework class library. The library will only be loaded if all three i.e. Matlab, PowerFactory and the wrapper are either 32 bit or 64 bit builds. It is important to note that due to limitations of .NET, Matlab has access to the values of an object and not the object itself. Figure 6 illustrates the coupling scheme.

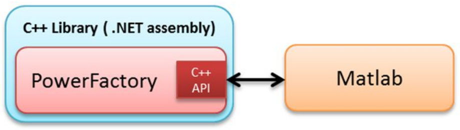


Fig. 6. PowerFactory / Matlab coupling using a .NET DLL (PF-ML-Coupler)

The Matlab example in Listing 1 shows how to activate a project in PowerFactory and run load flow. The C++ library is made visible to Matlab, after which the constructor is used to create an instance of PowerFactory. PowerFactory starts in engine mode and functions calls can be used to perform functions like changing the visibility of PowerFactory instance, active a project and run a load flow.

```
DllPath = 'C:\Users\LatifA\Desktop\PF-ML-Coupler.dll';
PFpath = ‘C:\Program Files (x86)\DIgSILENT\PowerFactory 15.1’
PFwrapperDll = NET.addAssembly(DllPath);
PFinstance  = PFnamespace.PowerFactory(PFusername, PFpassword,
PFpath);
PFinstance.setVisibility(isPFvisible);
PFinstance.ActivateProject(PFproject);
isLFvalid = PFinstance.RunLoadFlow(LFtype);
```

Listing. 1. Matlab load flow example using PF-ML-Couple

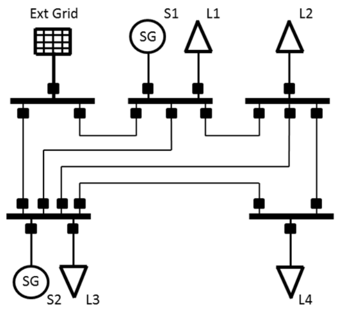
IV. Test Cases

In this section two implemented testcases will be dicussed. The testcases highlight the advantages and the possibilities one has by coupling the two simulators.

A. Optimizing custom objective functions

PowerFactory provides a tool for calculating optimal power flow. However, the objective functions are predefined and implementing user defined objective functions is yet not possible. Matlab is an ideal tool to implement optimization algorithm and define a custom objective function. Multi-objective optimization is another possibility.

Loss minimization (LM) is an optimization problem for scheduling generator output to minimize system losses. For the test case a simple five bus system has been implemented in PowerFactory with four lump loads, two generators and an external grid [6].



* The example has been taken from [6] example 7.9

Fig. 7. The 5 bus system

The LM problem has been formulated as a sum of losses of all transmission lines (TL), where P_{TL_i} and Q_{TL_i} are the active and reactive power loss of the ith TL, and S_l is sum of all TLs. N_l is the total number of TLs in the network.

$$F(S_l) = \sum_{i=1}^{N_l} \sqrt{({P_{TL_i}}^2 + {Q_{TL_i}}^2)}$$

The objective function can then be formulated as:

$$\min\,[\,F(S_l)]$$

Subject to constraints:

$$P_i^{min} \le P_i \le P_i^{max}$$

Where, P_i is the real power output of the ith generator and P_i^{min} and P_i^{max} are the lower and the upper bound of the ith generator. Listing 2 details the implementation of the objective function in Matlab.

```
function [ Fitness ] = CostFunction( P )
   global PFinstance;
   [r,c] = size(P);
   Fitness(r) = 0;
   for i = 1:r
      Pgini = P (i,1:c);
      PFinstance.SetPropertyDouble('Sync Gen 2','pgini',Pgini(1));
      PFinstance.SetPropertyDouble('Sync Gen 4','pgini',Pgini(2));
      PFinstance.RunLoadFlow(0);
      Fitness(i) = sqrt(PFinstance.GetPropertyDouble('Grid','c:LossP') ^2
       + PFinstance.GetPropertyDouble('Grid','c:LossQ') ^ 2);
   end
end
```

Listing. 2. Implementation of the objective function in Matlab

Table 1 lists the additional generation constraints used for the experiment.

TABLE I. GENERATOR OPERATIONAL LIMITS

Generator	$P^{min}(MW)$	$P^{max}(MW)$
S1	10	40
S2	20	100

Modified bat algorithm (MBA) [7] is a variant of bat algorithm; a population based metaheuristic optimization technique like particle swarm and genetic algorithm. The algorithm mimics the echolocation behavior most prominent in micro bats. The algorithm is an iterative algorithm, were in each new generation the bats converge towards the best experienced to that point and move away from the worst experiences.

In this experiment MBA has been used to optimize the formulated objective function. For the simulation, population size has been set to six and the maximum iteration limit has been set to 20. Figure 8 presents the simulation results. Optimal active power dispatch in this example is 40 MW and 88.87 MW for generator S1 and S2 respectively.

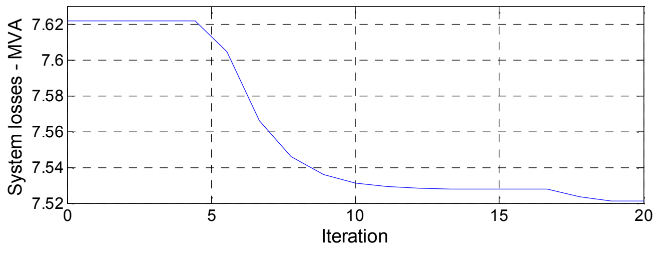


Fig. 8. Loss minimization using Modified Bat Algorithm

The experiment provides a proof of concept of optimizing user defined objective functions by coupling PowerFactory and Matlab. Matlab Optimization Toolbox provides a large number of algorithms which can alternatively be used to solve the problems with single and multiple objectives.

B. Implementing controller for steady state simulation

In areas with high PV penetration periods of high PV production and low consumer consumption results in over voltages. As the electrical distance from the transformer along the feeder increases this phenomenon becomes more prominent. In some cases the voltage rise can be severe enough to damage electrical equipment [8].

Variable power factor control (VPFC) is a voltage control method that reduces network losses compared to constant power factor method. Another advantage of using variable power factor control is that it is open loop hence very stable [9]. The German guide line VDE-AR-N 4105 also recommends characteristics shown in figure 12 [10]. Figure 9 illustrates controller design used in the experiment.

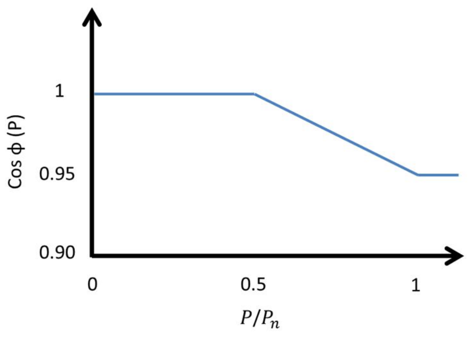


Fig. 9. Characteristics of the voltage controller

In the following example several different scenarios (without PVs, PVs with controllers and PVs with voltage controllers) are simulated. In the next step a list of variables to be logged is imported from text files after which, profiles are loaded from comma separated files and assigned to the individual loads and PVs. Steady state simulation is run after implementation of the VPFC and updating object properties from either controller output or the loaded profiles. The simulation ends when either EOF or invalid load flow is condition is true. The flow chart in figure 10 gives an overview of the simulation setup.

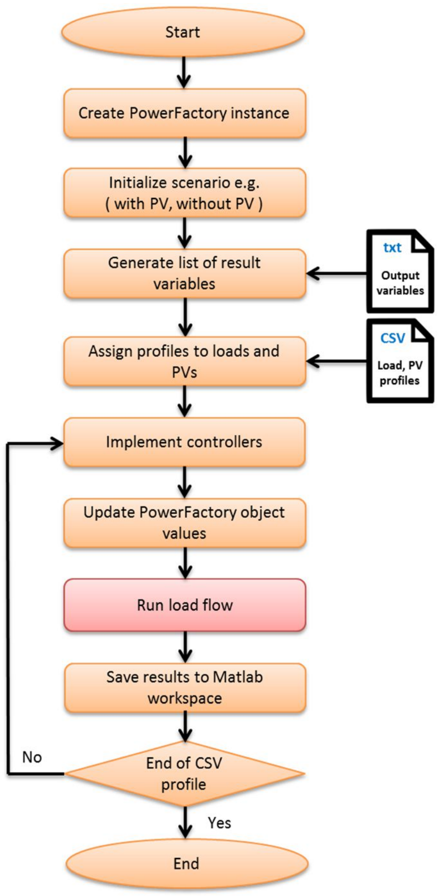

Start
Create PowerFactory instance
Initialize scenario e.g. (with PV, without PV)
Generate list of result variables
txt
Output variables
Assign profiles to loads and PVs
csv
Load, PV profiles
Implement controllers
Update PowerFactory object values
Run load flow
Save results to Matlab workspace
End of CSV profile
No
Yes
End

Fig. 10. Flow chart for test case 2

Nameplate PV output (S_Max) for the implemented example is 5 kVA for all the PVs in the distribution system. Reactive power limits (Q_Max) have been set to 1 kVAR. Listing 3 is an implementation of an open loop voltage controller in Matlab. The controller generates set points for the active power output and the power factor of the inverters subject to the inverter reactive power limits and the maximum possible power that can be generated.

```
function [ P,CosPhi ] = P_Phi_Controller_2(S,S_Max,Q_max)
   Snew = S/S_Max;
   if (Snew < 0.5)
      CosPhi = 1;
      P =S;
   elseif (Snew > 1)
      CosPhi = 0.95;
      P = Calculate_P(S,CosPhi,Q_max);
   else
      CosPhi = -0.1*Pnew+1.05;
      P = Calculate_P(S,CosPhi,Q_max);
   end
end

function [P] =  Calculate_P(S,CosPhi,Q_max)
   P = S * CosPhi;
   Q = S * sin(acos(CosPhi));
   if Q > Q_max
      P = Q_max/tan(acos(CosPhi));
   end
end
```

Listing. 3. Implementation of PV voltage controllers in Matlab

The graphs in the left column (figure 11) are the voltages measured at the point of common coupling (PCC) of the grid and the PVs. The effect of the controller on the voltage at PCC is clearly visible. When the inverter output increases above 50% of the nameplate power, reactive power compensation suppresses the voltage at PCC. Once the inverters reactive power capability is reached, the active power is curtailed to insure that the power factor is within the intended bounds.

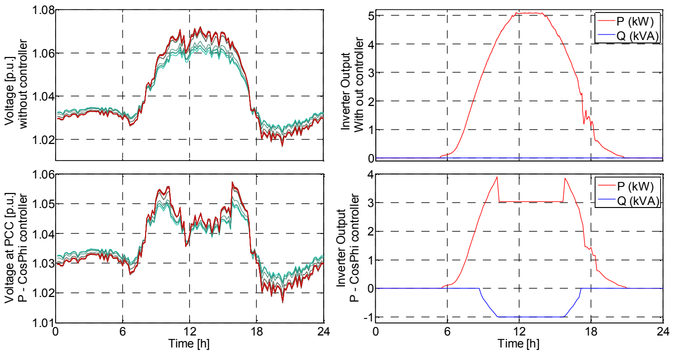


Fig. 11. Simulation results for test case 2

The test case provides an example for data logging, profile assignment and controller implementation. In similar fashion other controllers e.g. an Onload-tap-changer controller can easily be implemented in Matlab for steady state simulations.

V. Conclusion

In this section a conclusion is derived in light of the implemented test cases and a comparison with other coupling methods discussed in section II.

A. Ease of implementation

The single biggest advantage of the proposed coupling method is the ability to access both PowerFactory object parameters and internal calculation parameters directly from Matlab. In listing 2, internal PowerFactory calculations have been used to calculate the fitness of the objective function instead of exporting all line losses to Matlab, thereby reducing the number of parameters communicated to Matlab each time step.

The implemented PowerFactory API wrapper "PF-ML-Coupler" provides a number of high level functions that can called directly from Matlab. These functions can use to automate simulations with different scenarios and export results to Matlab. A proof of concept has been implemented in test case 2.

All PowerFactory power system analysis tools e.g. modal analysis, reliability assessment, protecting etc. can also be executed directly from Matlab.

One drawback of DPL scripting it does not provide support for code versioning. Moreover, in case of file sharing, one part of the code is implemented in PowerFactory and the other part is implemented in Matlab, which makes it difficult for a new user to understand the code. While working with the new library, a software repository tool can be used to version the code written in Matlab.

B. Flexibility

Matlab is a powerful scripting language that can be used to implement custom programs for many different applications such as solving user defined optimization problems, implementing a custom load shedding algorithm or implementing a controller. The API wrapper, "PF-ML-Coupler" is an efficient tool to implement such problems. As an example, a user defined optimization algorithm has been implemented successfully in test case 1.

Matlab has a plethora of toolboxes and a very large active community. These tool boxes can be used in conjunction with power system simulations to get a better insight of the problem. For example the statistics toolbox can be used to understand the extent of effectiveness of a demand side management algorithm when used under several hundred different scenarios.

It is further important to mention that presented wrapper can be extended to achieve the additional functionality. This is pretty straightforward due to the adopted methodology for coupling these two software packages.

C. Reduction in communication overhead

Schemes descripted in section II have inherent delay that result in a slower interfacing for example file based exchange it is obvious that file access is the slowest possible interconnection. For the build in DSL Matlab interface it comes from invoking Matlab in every time step, need for creating global variables in case of using Simulink and re-initializing it with persisting variables for every time step

REFERENCES

[1] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp and L. Mili, "Power system and communication network co-simulation for smart grid applications," in *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, 2011.

[2] D. Powerfactory, "PowerFactory User's Manual," *DIgSILENT, GmbH,* 2011.

[3] M. U. Guide, "The mathworks," *Inc., Natick, MA,* vol. 5, p. 333, 1998.

[4] A. Stativa, M. Gavrilas and V. Stahie, "Optimal tuning and placement of power system stabilizer using particle swarm optimization algorithm," in *Electrical and Power Engineering (EPE), 2012 International Conference and Exposition on*, 2012.

[5] M. Stifter, R. Schwalbe, F. Andr{\'e}n and T. Strasser, "Steady-state co-simulation with PowerFactory," in *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2013 Workshop on*, 2013.

[6] H. Saadat, *Power System Analysis,(2nd),* McGraw-Hill Higher Education, 2009.

[7] A. Latif and P. Palensky, "Economic Dispatch Using Modified Bat Algorithm," *Algorithms,* vol. 7, no. 3, pp. 328-338, 2014.

[8] R. Passey, T. Spooner, I. MacGill, M. Watt and K. Syngellakis, "The potential impacts of grid-connected distributed generation and how to address them: A review of technical and non-technical factors," *Energy Policy,* vol. 39, no. 10, pp. 6280-6290, 2011.

[9] E. Demirok, P. Casado Gonzalez, K. H. Frederiksen, D. Sera, P. Rodriguez and R. Teodorescu, "Local reactive power control methods for overvoltage prevention of distributed solar inverters in low-voltage grids," *Photovoltaics, IEEE Journal of,* vol. 1, no. 2, pp. 174-182, 2011.

[10] V. V. d. E. E. Informationstechnik, "eV: VDE-AR-N 4105: 2011-08: Power generation systems connected to the low-voltage distribution network Technical minimum requirements for the connection to and parallel operation with low-voltage distribution networks," *English translation of the VDE application rule VDEAR-N-4105.*