Electric Power Exchanges with Sensitivity Matrices An Experimental Analysis

Martin Drozda[1] [2]

Los Alamos National Laboratory[3]
Los Alamos, POBOX 1663, MS M997, NM 87545

Abstract

We describe a *fast incremental* method for power flows computation tailored for electric power markets. Fast in the sense that it can be used for *real time* power flows computation, and incremental in the sense that it computes any *additional* increase/decrease in line congestion caused by a particular (n-lateral) contract. Incremental methods offer a powerful way of dealing with congestion contingency, improve information flows among market entities, and are easy to understand. In this document we provide basic scaling properties for electric power markets using this method, and compare it with methods using (linearized) power flow code.

The author is in the process of obtaining a patent on methods, algorithms, and procedures described in this paper.

1 Introduction

Deregulated electric power market assigns new responsibilities and tasks to market entities. The task of system operator is to keep power flows in an electric grid within limits. The task of electric power exchange is to couple power producers and consumers. Furthermore, power exchange has to communicate with system operator to ensure stability of the grid after contracted power is injected at a generator bus, and taken out at a load bus. This simple interaction schema encompasses a multitude of problems that were subdued when the centralized utility companies ruled the electricity world. Nowadays, a power exchange has to face tens of thousand of contracts seeking settlement each hour. For each contract it needs to be decided whether the contract is feasible, i.e., complying with security guidelines of the power grid. This is done by system operator, and ideally, he should run a power flow code, and consider projected power flows. This requires a fast method of power flow computation. Furthermore, additional power injection is going to influence market situation. Increase/decrease in line congestion forces consumers to buy power from different producers as they have to obey the very same security guidelines. The power exchange usually tries to intervene and designs a set of rules aimed to resolve congestion. This is done by charging for additional congestion caused by a contract. Several schemata were suggested for congestion fee computation. We can divide them into three groups: contract path, point-to-point, and real flow. The real flow methods ensure the fairest pricing but are computationally very expensive. Some simplifications were proposed, e.g., flow gates [8]. In this document we describe a fast incremental method for power flows computation. Fast in the sense that it can be used for real time power flows computation, and incremental in the sense that it computes any additional increase/decrease in line congestion caused by a particular (n-lateral) contract.

Our method is based on sensitivity analysis of the underlying power grid. For the analysis we used a standard linearized power flow code. Let n be number of buses, and m number of lines. We sequentially inject a unit amount of power into each bus, and compute the increase in power flows on

[1]Author visiting from Slovak Academy of Sciences, Institute of Control Theory and Robotics, Slovak Republic.

[2]`drozda@lanl.gov`

[3]The Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; therefore, the Laboratory as an institution does not endorse the viewpoint of a publication or guarantee its technical correctness. Research supported by the US Department of Energy under contract W-7405-ENG.

each line. This way we obtain a sensitivity matrix of $n \times m$ cells. The matrix can also be computed analytically. The matrix is static, and needs to be re-computed each time physical parameters of the power grid change. This problem has been addressed by a distinct paper [6] from our team, and we can re-compute sensitivity matrices for a moderate grid size, e.g., ERCOT within a few minutes (1-2). Computation of incremental power flows is achieved by a single multiplication with injection vector. Injection vectors for bilateral contracts have the property of being very sparse, thus, the multiplication reduces to multiplication/addition of four numeric values.

Our experiments show that we speed up contract approval/disapproval by system operator significantly. Computational time grows linearly in the number of lines, whereas in case of a standard linearized power flow code the computational time grows polynomially. This for example means that in case of ERCOT we were able to speed up the process more than 120 times. With this method we were able to simulate markets of ERCOT or WSCC on a single PC (500MHz, 256MB). Moreover, we are able not only approve/disapprove contract in real time, but also compute increase/decrease in line congestion caused by the contract, thus, giving data needed by power exchange for congestion management.

This document is organized as follows: in Section 2 we give an overview of the market implemented in ELISIMS, a software tool for simulating electric power markets; in Section 3 we give an insight into the incremental method, and we show how it is used by system operator to approve/disapprove contracts; in Section 4 we show basic scaling properties of electric power exchanges using this method, and compare it with electric power exchanges using (linearized) power flow code to compute power flows.

2 The market

For our simulation tool (ELISIMS [3]) we have chosen to implement the so called *continuous nodal market*. It is a market with nodal prices as opposed to markets with zone prices. We have designed the market as a 24 hour forward market with no possibility of spot trading. The market allows only bilateral contracts which are processed in the order they come. The electric power price at each node is a function of the electric power price for the given hour and given generator, and transmission cost. The transmission costs are based on congestion. In early versions of ELISIMS we have implemented congestion fee computation based on *contract paths*. At present, we use *real-flow* computation methods based on sensitivity matrices. The contract paths were computed as the shortest path from a generator to a customer. To compute the shortest path we used Dijkstra's shortest path algorithm (see [2]).

Next, we describe mechanisms of our market to make you accustomed with the interplay among all the market entities: customers, power producers (generators), system operator, and power exchange. System operator is an entity which is responsible for stability and security of the grid (under cooperation with transmission operators and owners); it runs the network from the physical point of view, i.e., it operates buses, branches, controls grid network limits etc. It does not have any obligations against customers, producers, or brokers and actually it should be completely independent from all other entities. On request from the power exchange gives answers on whether a new contract is going to preserve all security and stability parameters of the network. To cope with this task, the system operator runs a power flow code. Based on the projected power flows, he reasons about future security conditions on the power grid. The power flows computation is done in real time. There are tens of thousand of contracts awaiting approval each hour. However, it is not only approval or disapproval of contracts that are in the scope of his responsibilities. Markets entities need strong hints in order to modify a disapproved contract, and as well, they need to understand trends in power grid congestion.

Power exchange is an entity that runs the market itself. It collects orders from consumers, producers, and brokers and tries to clear them. Congestion management is an important factor in the process of market functioning. Congestion changes behavior of everyone. The once cheap power becomes untransportable from the point of production, and consumers are forced to seek alternative power producers. On the other hand, producers cannot sell their power to an arbitrary consumer due to line congestion. The power exchange intervenes in this situation, and designs a set of rules to control congestion. This is mostly done by charging fees for any additional congestion that a contract may cause. For an exact and fair congestion fee computation, the power exchange needs closely cooperate with the system operator. The system operator has strong tools to provide the power exchange with all necessary information. He runs a power flow code for each contract clearing. The system operator also needs to supply the power exchange with incremental data about changing line congestion. In other words, he needs to run a power flow code that computes incremental power flows, and the computation needs to be doable in real time.

Now, suppose you are a customer willing to buy some power to satisfy your needs. You submit your request to the power exchange. The power exchange finds the cheapest power producers from the point of view of power price, and congestion fees connected to the contract approval. The power exchange cooperates with system operator on contract approval, and seeks approval. The system operator either approves or disapproves the contract, and supplies the power exchange with incremental data about line congestion. This data is used to compute real-flow congestion fees, and adjusts the approximate congestion fees computed by power exchange in order to find the cheapest producer.

ELISIMS is written in ANSI/ISO C/C++ for UNIX/Linux environment. The tool is capable of using two kinds of power flow codes. First of the two is the power flow code developed at the University of Texas at Arlington, which is a well tested non-linear power flow code. The other one is a linearized version of power flow code developed at the Los Alamos National Laboratory. Due to the necessity to start the simulation from zero loads and power generation (i.e. black start), we found the linearized version easier to use. The computational error of the linearized version is, in the case of electric power market, acceptable.

We note that running power flow code of any kind is the most computational time consuming procedure in our simulation. It can consume 85–95% of used processor time. In our case the used hardware was a PC based on an Intel Pentium III 500 MHz processor and 256MB of memory. Running the simulation on the ERCOT network with the linearized power flow code took us 2 hrs 12 mins to simulate one real-time hour[4] (i.e. 52 hrs 48 mins to run the whole 24 hrs market). 88% of this time was spent to run the power flow code. [5] contains basic results on scalability of electric power exchanges using linearized power flow code.

[4]Parameters of the network were: topology = ERCOT, number of buses = 4,527, size of contracts = 6.2MW, capacity of branches = ERCOT, demand/power available ratio = 0.86. Number of branches for ERCOT is 5,412. Number of contracts necessary to settle one real time hour was approximately 5,700.

3 Sensitivity matrix

Let us have a power network of n buses, and l lines. Let $q = [q_1, ..., q_n]^T$ to be an injection vector, and let $limit = [limit_1, limit_2, ..., limit_l]$ to be a vector of *residual* line limits. Residual line limits represent amount of remaining capacity on each line.

The sensitivity matrix of the underlying system is an $n \times l$ matrix. Let i be an arbitrary bus. Then the i-th column of the matrix represents the *incremental change* on each line caused by injection of a unit amount of power into the i-th bus.

Let us now assume that there is a contract that needs to get evaluated with respect to the limits of a power grid. Let the contract to be a bilateral contract. In this case the injection vector is going to be a sparse vector of two entries. One for the generator bus i, and one for the load bus j. To check against the limits of the power grid we need to recall columns i and j from the sensitivity matrix. Let us assign s_i and s_j to columns i and j, respectively, i.e., s_i and s_j are now *injection sensitivity vectors* for bus i and j. In the case of a bilateral contract (single generator, single consumer) the following inequality has to be satisfied to preserve line limits in a power grid:

$$q \times (s_i + s_j) \leq limit.$$

This can be generalized for multilateral contracts. Let the number of parties in such a contract to be b. The injection vector q becomes a vector of b non-zero entries, and the inequality becomes:

$$q \times \sum_{i=1}^{b} s_i \leq limit.$$

The left size of the inequality represents incremental change of line congestion caused by the b-lateral contract.

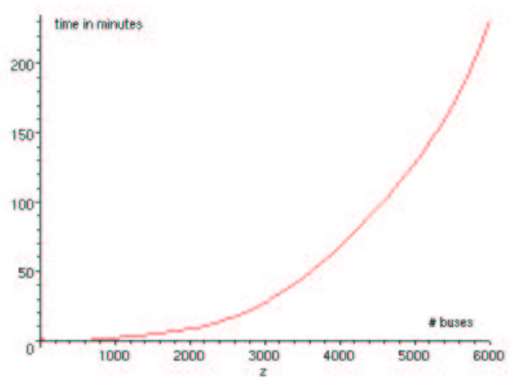

Figure 1: Scaling of computation time needed to compute sensitivity matrices with respect to number of buses.

4 Experiment

In this section we present results that we got running ELISIMS with sensitivity matrices.

4.1 Computing sensitivity matrices

In the previous section we gave a definition of sensitivity matrix. Now, we show how to compute sensitivity matrices efficiently on any desktop PC computer. For this purpose we used a linearized version of power flow code. Hence, we prefer a heuristic method to a direct analytical method. In our approach we start with a power network with no power injection on either generator or consumer side. We assume a lossless power grid. Now, we sequentially and separately inject 1MW of power into each bus. We follow the sign convention where generator injections have positive value, and vice-versa load buses (consumers) out-flows (consumptions) have negative values. After injecting 1MW into a given bus we run the linearized power flow code and analyze power flows that this injection caused. By doing so for each bus separately we obtain a sensitivity matrix of dimension $n \times l$, where n is number of buses and l is number of lines (branches).

We have implemented this algorithm and ran numerous experiments. Our hardware was a PC (Intel III 500MHz, 256MB RAM, Linux). In figure 1 we depict scaling characteristics for computation of sensitivity matrices. We have done this for a random network of 1,000–6,000 buses. We were unable to compute sensitivity matrices for networks with a larger number of buses on a (single-processor) PC with limited memory. Nevertheless, this gives a good idea how much time one will need to compute a sensitivity matrix. Under *random network* we understand a tree graph with random degree (2–5) for each node and 10% edges (randomly) added among nodes. This random addition of edges makes the graph (network) to be a non-tree graph. We have used this kind of network in our previous paper [5]. It gives a quite conservative image of real power networks, and all measurable properties fluctuate at their upper bounds. However, a closer look at real network unveils that these are almost tree structures with a tangle of additional lines around bigger conglomerations.

One can argue that use of sensitivity matrices can be very cumbersome because of the need to recompute them at every change of physical parameters of power network. However, this does not happen so frequently, and in case of scheduled maintenance the system operator can get prepared well ahead. Nevertheless, we have designed a distributed method for computation of sensitivity matrices with much more favorable speed and scaling characteristics, see [6]. This method gives a possibility to recompute a sensitivity matrix for a power grid of size of ERCOT, or WSCC within a few minutes. A more serious problem is the size of memory needed to store sensitivity matrices. Indeed, if we imagine a huge power network such as the one in the US with projected number of customers reaching 150 million this can be a considerable problem. Fortunately, the number of lines is going to be relatively stabilized. Use of local methods for computation and look-up of injection sensitivity vectors can be probably very successfully used. At present, we can store a sensitivity matrix for ERCOT or WSCC on a desktop PC with a few gigabytes of memory – without exploring a distributed memory, or other approaches.

4.2 Running time of an electric power exchange using sensitivity matrix

In this section, we will present two experiments: the first one is running an electric power exchange using a linearized power flow code, and the other one is running using the underlying sensitivity matrix. We have taken the first experiment from our previous paper [5] which discusses approaches using linearized power flow codes in more detail.

To reason about efficiency of sensitivity matrices we

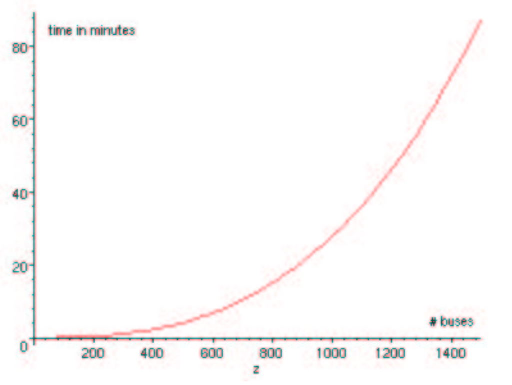

Figure 2: Scaling of an electric power exchange with respect to number of buses using a linearized power flow code.

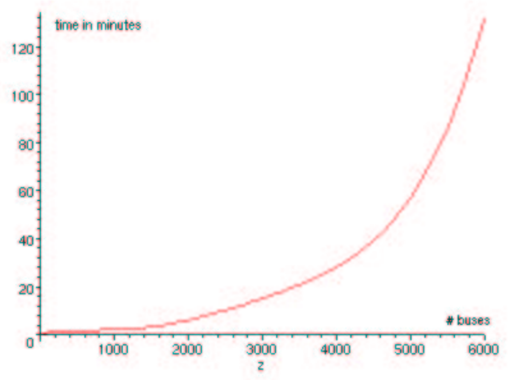

Figure 3: Scaling of an electric power exchange with respect to number of buses using underlying sensitivity matrix.

have decided to run a series of simulations. The only independent variable we use is number of buses, and the only dependent variable we use is time within which an electric power exchange clears all contracts for a 24-hour period (single business day). In Figure 2 we depict this situation with use of a linearized power flow code, and in Figure 3 we do the same for the sensitivity matrix.

We can clearly see that sensitivity matrices offer a reasonable speed-up compared to the approach using linearized power flow code. The networks that we have used in these experiments are the random networks described in previous subsection. As we have already mentioned these represent a quite conservative view of real power networks. We have run this experiment also for ERCOT, the power network of the state of Texas. ERCOT is a network of with 4,527 buses, and 5,412 power lines. With a linearized power flow code we were able to run a single business day in 52 hrs 48 mins, and with the underlying sensitivity matrix this time shrunk to mere 30 minutes. This does not come as a surprise because from previous experiments described in [5] we know that roughly 88% of computation time is spent in running power flow code. Cutting down on time spent in this part of the code one can extremely speed up the whole clearing process what leads to an increased reliability of the whole process and eases up pressure on the system operator.

5 Conclusions

In the paper we have presented an incremental approach to power flows computation tailored for electric power exchanges. We have run a series of experiments to be able to reason about efficiency of this approach. We have found out that using sensitivity matrices in the process of clearing an electric power market can lead to a significant speed-up and thus to improved reliability of the electric grid. In case of ERCOT, the power network of the state of Texas, we have managed to cut running time from almost 53 hrs down to 30 minutes. We consider this speed-up worth of notice and future research.

The weakness of sensitivity matrices stems from the problem of their storage. They create densely populated matrix of injection sensitivity vectors. In the case we have to deal with 150 million customers the matrix can be very big, though, the other dimension which is number of lines will probably stay fixed or will only slowly increase in the future. We believe that use of local methods can ease up this problem. Also, distributed memory storage methods can be considered. However, in most practical cases, one will only need to store a network of size of ERCOT or WSCC what is possible on a single PC with a few gigabytes of memory.

The advantage of sensitivity matrices is that they can become public data accessible to anyone connected to electric power business. They increase information flows and can significantly reduce number of unnecessary contract approval requests submitted to the system operator. With sensitivity matrices each entity of power market will be able to evaluate its own business decisions before they are made, and the decision helper software will be run on a PC based computer.

Moreover, this method of power flows computation is inherently incremental. We consider this feature to be a big advantage over other methods as it positively contributes to congestion management.

References

[1] R. Bohn, M. Caramanis and F. Schweppe. Optimal Pricing in Electrical Networks Over Space and Time. *Rand J. on Economics*, 18(3), 1984.

[2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[3] L. J. Dowell, M. Drozda, D. B. Henderson, V. W. Loose, M. V. Marathe, and D. J. Roberts. *Electric Industry Simulation System (ELISIM)*. Los Alamos National Laboratory, Technical Rep. No. LA-UR-00-1572.

[4] L. J. Dowell, M. Drozda, D. B. Henderson, V. W. Loose, M. V. Marathe, and D. J. Roberts. *ELISIMS: Comprehensive Detailed Simulation of the Electric Power Industry*. Los Alamos National Laboratory, Technical Rep. No. LA-UR-98-1739.

[5] L. J. Dowell, M. Drozda, D. B. Henderson, V. W. Loose, M. V. Marathe, D. J. Roberts. *Scalability of ELISIMS: Comprehensive detailed simulation of the electric power industry*. IEEE SMC 2000 Conference, Nashville, Tennessee, October 8-11, 2000.

[6] L. Jonathan Dowell, Douglas J. Roberts, Dale B. Henderson. *On Solving Nearly-Singular, Sparse Systems of Linear Equations: Diakoptics Techniques for Parallel Computing*. Los Alamos National Laboratory, Technical Rep. No. LA-UR-00-2175.

[7] W. Hogan. Contract networks for electric power transmission. *J. Regulatory Economics,* pp. 211-242, 1992.

[8] W. Hogan. *Flowgate Rights and Wrongs*. ksghome.harvard.edu/∼.whogan.cbg.ksg/

[9] S. Pissanetsky. *Sparse matrix technology*. Academic Press, London, 1984.

[10] F. A. Wolak *An Empirical Analysis of the Impact of Hedge Contracts on Bidding Behavior in a Competitive Electricity Market*. www.stanford.edu/∼wolak/

[11] F. F. Wu, and P. Varaiya. *Coordinated Multilateral Trades for Electric Power Networks: Theory and Implementation*. www.path.berkeley.edu/∼varaiya/power.html

[12] The Changing Structure of the Electric Power Industry: Selected Issues. DOE 0562(98), Energy Information Administration, US Department of Energy, Washington, D.C. 1998.