Scalable Newton-Krylov Solver for Very Large Power Flow Problems

Reijer Idema, *Student Member, IEEE*, Domenico J. P. Lahaye, Cornelis Vuik, and Lou van der Sluis, *Senior Member, IEEE*

***Abstract*—The power flow problem is generally solved by the Newton-Raphson method with a sparse direct solver for the linear system of equations in each iteration. In this paper, alternatives based on iterative linear solvers are presented that are faster and scale much better in the problem size, making them ready for the ever-growing power systems of the future. For the largest test problem, with around one million busses, the presented alternative is over 120 times faster than using a direct solver.**

***Index Terms*—Approximate minimum degree reordering, ILU *k*-levels, Krylov methods, LU factorization, Newton-Raphson method, power flow analysis, preconditioning, scaling.**

I. Introduction

The power flow, or load flow, computation is the most important network computation in power systems. It calculates the voltage magnitude and angle in each bus of a power system, under specified system operation conditions. Other quantities, such as current values, power values, and power losses, can be calculated easily when the bus voltages are known. In other words, power flow computations provide a steady-state simulation of power systems.

The consumption of electricity keeps on rising each year. As a result, power systems grow larger and more complex to supply all consumers.

More and more nationwide power systems are being connected to each other, to be able to exchange cheap excess power. This results in huge connected power systems, with many times the busses and transmission lines of the classical systems. A small set of simultaneous failures could propagate through the entire system, causing a massive blackout. Therefore, providing security against overloading is more important than ever.

Another important development in power systems is the incorporation of renewable energy sources, such as wind and solar energy. Traditional generators are connected to the transmission network. This is referred to as centralized power generation. Renewable energy generation is often decentralized, i.e., connected to the distribution network at consumer level. Also, renewable sources are mostly natural and often have an uncontrollable power output.

Power flow calculations are generally done on the transmission network, and the distribution network is aggregated at busses in the power system model. The decentralized nature of the renewable energy generation, however, may in time call for a more complex distribution network that can no longer easily be aggregated. Then power flow calculations may also have to incorporate the distribution network, resulting in power flow problems of sheer massive size.

Traditionally the power flow problem is solved by use of the Newton-Raphson method, with a direct solver for the linear systems [1], [2]. It has been recognized that iterative linear solvers can offer advantages over sparse direct solvers for large power systems [3]–[7].

The question arises when iterative methods are better than direct methods for power flow problems. The key to answering this question is in the scaling of the computational time of the power flow algorithm in the problem size.

This question is answered in this paper for a test set of power flow problems ranging up to one million busses. The scaling of the Newton-Raphson method with a direct solver is tested, and it is compared with the scaling behavior of the Newton-Raphson method with an iterative linear solver with a selection of preconditioners.

We show that direct solvers, and other methods using a complete LU factorization, scale very badly in the problem size. The alternatives proposed in this paper are faster for all tested problems and show near linear scaling, thus being much faster for large power flow problems. Using a direct solver the largest problem takes over an hour to solve, while our solver can solve it in less than 30 s, that is 120 times faster.

Furthermore, the proposed methods have more variables to tune than when using a direct linear solver. As such, they offer more options to reuse information and tweak settings when solving many closely related power flow problems, as is done for example in contingency analysis.

The implementation of our solver is done in C++ using Portable, Extensible Toolkit for Scientific Computation (PETSc) [8], a state-of-the-art C library for scientific computing. PETSc can be used to produce both sequential programs, and programs running in parallel on multiple processors. The experiments of which the results are presented in this paper were all executed on a single processor core.

The test set of power flow problems used is based on the UCTE[1] winter 2008 study model, which consists of 4253 busses

Manuscript received January 11, 2011; revised May 03, 2011, June 17, 2011, and August 12, 2011; accepted August 15, 2011. Date of publication September 22, 2011; date of current version January 20, 2012. Paper no. TPWRS-00020-2011.

R. Idema, D. J. P. Lahaye, and C. Vuik are with the Delft Institute of Applied Mathematics, Delft University of Technology, Delft, The Netherlands.

L. van der Sluis is with the Power Systems Department, Delft University of Technology, Delft, The Netherlands.

Digital Object Identifier 10.1109/TPWRS.2011.2165860

[1]UCTE is a former association of transmission system operators in Europe. As of July 2009, the European Network of Transmission System Operators for Electricity (ENTSO-E), a newly formed association of 42 TSOs from 34 countries in Europe, has taken over all operational tasks of the existing European TSO associations, including UCTE. See http://www.entsoe.eu/

and 7191 lines. The smallest problem is the UCTE winter 2008 study model itself, while the larger problems are all constructed by copying the model multiple times, and interconnecting the copies with new transmission lines.

II. Power Flow Problem

The power flow problem is the problem to determine the voltage at each bus of a power system, given the supply at each generator and the demand at each load in the network.

Let $Y = G + jB$ denote the network admittance matrix of the power system. Then the power flow problem can be formulated as the nonlinear system of equations

$$\sum_{k=1}^{N} |V_i||V_k|(G_{ik}\cos\delta_{ik} + B_{ik}\sin\delta_{ik}) = P_i \quad (1)$$

$$\sum_{k=1}^{N} |V_i||V_k|(G_{ik}\sin\delta_{ik} - B_{ik}\cos\delta_{ik}) = Q_i \quad (2)$$

where $|V_i|$ is the voltage magnitude, δ_i is the voltage angle, with $\delta_{ij} = \delta_i - \delta_j$, P_i is the active power, and Q_i is the reactive power at bus i. For details, see for example [9] and [10].

Define the power mismatch function as

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} P_i - \sum_{k=1}^{N} |V_i||V_k|(G_{ik}\cos\delta_{ik} + B_{ik}\sin\delta_{ik}) \\ Q_i - \sum_{k=1}^{N} |V_i||V_k|(G_{ik}\sin\delta_{ik} - B_{ik}\cos\delta_{ik}) \end{bmatrix} \quad (3)$$

where $\boldsymbol{x}$ is the vector of voltage angles and magnitudes. Then the power flow problem (1), (2) can be reformulated as finding a solution vector $\boldsymbol{x}$ such that

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}. \quad (4)$$

This is the system of nonlinear equations that we solve to find the solution of the power flow problem.

III. Power Flow Solution

Our solver is based on the Newton-Raphson method for systems of nonlinear equations like (4). This iterative method updates the approximate solution $\boldsymbol{x}_i$ in iteration i with a Newton step $\boldsymbol{s}_i$, calculated from the linear system of equations

$$J_i\boldsymbol{s}_i = -\boldsymbol{F}_i \quad (5)$$

where J_i is the Jacobian of the power mismatch $\boldsymbol{F}$. For details on the Newton-Raphson method, see for example [11].

In power flow analysis, the classical way to solve the linear system (5) is with a direct solver. Effectively, this means that J_i is factorized into a lower triangular factor L_i, and an upper triangular factor U_i, such that $L_iU_i = J_i$. Then the linear problem $L_iU_i\boldsymbol{s}_i = -\boldsymbol{F}_i$ can be solved using forward and backward substitution, which are both very fast operations.

The LU factorization was originally designed for full matrices, and has complexity n^3 in the problem size n. It was adapted very efficiently for sparse matrices, like the Jacobian matrix J_i in the power flow problem. Although the scaling of the LU factorization's computational time in the problem size is much more complex for sparse matrices, it is well-known that in general it scales worse than linearly for large problem sizes. This is also clearly illustrated by the numerical experiments in Section VII. For more information on sparse direct methods, see for example [12] and [13].

To get a power flow solver that scales better for large problems, we look at alternative linear solvers for problem (5) in the form of iterative methods, in particular the Generalized Minimal Residual method (GMRES) [14]. The reason to use GMRES is that it, being a minimal residual method, solves the problem in the least number of matrix-vector multiplications. Whether other iterative linear solvers, like Bi-CGSTAB [15], [16] or IDR(s) [17], should also be considered, can be derived from the performance of GMRES.

With the Newton-Raphson method, there is only a certain accuracy that can be reached in each iteration. Solving the linear system to an accuracy higher than that needed to reach the best Newton-Raphson accuracy in the current step would be a waste of computational time. Therefore, when using an iterative linear solver, the linear systems should not be solved to full precision. Instead forcing terms η_i should be determined, and the iterative linear solver used to solve up to

$$\|J_i\boldsymbol{s}_i + \boldsymbol{F}_i\| \leq \eta_i\|\boldsymbol{F}_i\|. \quad (6)$$

This is called an inexact Newton-Krylov method [18].

In [6], we discussed three methods of choosing the forcing terms η_i. Throughout this paper, the method proposed by Eisenstat and Walker [19] is used. This method has been successfully used in practice on many different problems, and also provided very good results for us. The method based on [20] gave similar good results for our test cases, but the method derived in [21] generally yielded forcing terms that were too small, resulting in oversolving.

IV. Preconditioning

Essential to the performance of a Krylov method like GMRES is a good preconditioner. See for example [22] for information on preconditioning. In our solver, we use right preconditioning, meaning that we iteratively solve the linear system

$$J_iP_i^{-1}\boldsymbol{z}_i = -\boldsymbol{F}_i \quad (7)$$

and then get the Newton step $\boldsymbol{s}_i$ by solving $P_i\boldsymbol{s}_i = \boldsymbol{z}_i$. For fast convergence, the preconditioner matrix P_i should resemble the coefficient matrix J_i. At the same time, a fast way to solve linear systems of the form $P_i\boldsymbol{u}_i = \boldsymbol{v}_i$ is needed, as such a system has to be solved in each iteration of the linear solver.

In this paper, we consider preconditioners in the form of a product of a lower and upper triangular matrix $P_i = L_iU_i$. Any linear system with coefficient matrix P_i can then simply be solved using forward and backward substitution. To get such a preconditioner, we choose a target matrix Q_i and then construct either the LU factorization $L_iU_i = Q_i$ or an incomplete LU (ILU) factorization [23], [24] L_iU_i of Q_i. In the case of the full LU factorization, this leads to a preconditioner $P_i = L_iU_i = Q_i$, whereas with the ILU factorization, the preconditioner $P_i = L_iU_i$ only resembles the target matrix Q_i. The trade-off here is that an ILU factorization is cheaper to build than an LU factorization, whereas the full LU factorization will generally result in a better preconditioner.

There are three choices that are considered for the target matrix Q_i. These are the Jacobian matrix $Q_i = J_i$, the initial Jacobian matrix $Q_i = J_0$, and $Q_i = \Phi$, where

$$\Phi = \begin{bmatrix} B' & 0 \\ 0 & B'' \end{bmatrix} \tag{8}$$

with B' and B'' as in the BX scheme of the Fast Decoupled Load Flow (FDLF) [25] method as proposed by van Amerongen [26].

The FDLF matrix Φ can be seen as an approximate Schur complement of the initial Jacobian matrix [27]. In previous studies, Φ has already proven to be a good preconditioner [3], [6], while containing only half the non-zeros of the Jacobian matrix, thus providing benefits in computing time and memory.

Other preconditioners that are known to often work well for large problems are preconditioners based on iterative methods. Only stationary iterative methods can be used as a preconditioner for standard implementations of GMRES, Bi-CGSTAB, and $\mathrm{IDR}(s)$. Non-stationary iterative methods, like GMRES itself, can only be used with special flexible iterative methods, like FGMRES [28]. The use of FGMRES with a GMRES-based preconditioner has been explored in [7].

Algebraic Multigrid (AMG) methods can also be used as a preconditioner. Running one cycle of AMG, with a stationary solver on the coarsest grid, leads to a stationary preconditioner. Such a preconditioner is very well suited for extremely large problems. For more information on AMG, see [29, App. A].

V. Factorization

As mentioned in the previous section, complete and incomplete LU factorizations of the target matrix Q_i are used to construct preconditioners. In this section, we consider the quality and fill, i.e., the number of non-zeros, of the preconditioners and how to control these properties.

When using a full LU factorization, the quality of the preconditioner is predetermined by the choice of the target matrix, as $P_i = Q_i$. However, when doing an LU decomposition on a sparse matrix, the factors L_i and U_i generally have more non-zeros than the original matrix Q_i. This is referred to as fill-in. The number of non-zeros in the factors divided by the number of non-zeros in the original matrix is called the fill-in ratio. The higher the fill-in ratio is, the more memory and computational time is needed for the factorization and the forward and backward substitutions. It is therefore paramount to try to minimize the fill-in of the LU factorization.

The fill-in can be controlled by reordering the rows and columns of the matrix that has to be factored. However, finding the ordering that minimizes fill-in has been proven to be NP-hard [30]. Many methods have been developed to quickly find a good reordering; see for example [12] and [13].

The method of incomplete LU factorization used is $\mathrm{ILU}(k)$, where k is the number of levels of fill-in. This method determines the allowed non-zero positions in the factors based on the k-level, and subsequently calculates the best values for these non-zero positions.

As the fill-in is predetermined, one might think that reordering is not important. However, even if the reordering does not influence the fill-in for an $\mathrm{ILU}(k)$ factorization, it does influence the quality with which L_iU_i approximates Q_i.

TABLE I
LU Factorization of J_0 for uctew032

reordering	none	AMD
factorization time	33.6	0.53
fill-in ratio	35	2.3

TABLE II
Power Flow Test Problems

	busses	lines	nnz(J)
uctew001	4.25k	7.19k	62.7k
uctew002	8.51k	14.4k	125k
uctew004	17.0k	28.8k	251k
uctew008	34.0k	57.6k	502k
uctew016	68.0k	115k	1.00M
uctew032	136k	231k	2.01M
uctew064	272k	462k	4.02M
uctew128	544k	924k	8.05M
uctew256	1.09M	1.85M	16.1M

In our experiments, using no reordering was compared with the PETSc implementations of Nested Dissection (ND), One-way Dissection (1WD), Reverse Cuthill-McKee (RCM), Quotient Minimum Degree (QMD), and Approximate Minimum Degree (AMD). Of these, the AMD reordering [31] was a clear winner, being the fastest to compute while yielding the least fill-in with a full LU factorization and the best quality ILU factorizations at the same time. Reordering methods of UMFPACK [32], SuperLU [33], SuperLU_DIST [34], and MUMPS [35] were also tested, but none yielded an improvement over the PETSc AMD reordering for our problems.

To illustrate the impact of the AMD reordering, Table I shows the results of a single LU factorization of the initial Jacobian matrix J_0 for the uctew032 problem (see Table II). The factorization time is measured in seconds.

The influence of reordering on the $\mathrm{ILU}(k)$ factorization is less drastic, but still very useful. For the uctew032 case, when solving the initial Jacobian system $J_0\boldsymbol{s}_0 = -\boldsymbol{F}_0$ with an $\mathrm{ILU}(k)$ factorization of J_0 as preconditioner, with different choices of k, GMRES needed around 25% less iterations to converge when using the AMD reordering, compared to using no reordering.

VI. Flexibility

In the previous sections, a number of options were discussed, that have to be considered when solving a power flow problem with an inexact Newton method with preconditioned linear solver. These options do not only provide flexibility when dealing with a single power flow problem, but even more so when solving many closely related power flow problems, as for instance in contingency analysis.

For example, after solving a base case, a preconditioner can be made by factorizing the Jacobian in the solution of the base case. This preconditioner can then be used for all the derived power flow cases, saving time in the solution process of each of those cases.

Further, when solving a high amount of similar problems, some extra time can be spent on tailoring the preconditioning and forcing terms to the base case, because a lot of time can be won overall if each of the derived cases solves a bit faster.

TABLE III
PRECONDITIONERS

target	reordering	factorizations				fig.
J_i	AMD	LU	ILU(4)	ILU(8)	ILU(12)	1
J_0	AMD	LU	ILU(4)	ILU(8)	ILU(12)	2
Φ	AMD	LU	ILU(4)	ILU(8)	ILU(12)	3

TABLE IV
SOLUTION TIME FOR SMALL TEST CASES

busses	4.25k	8.5k	17k	34k
direct solver	0.077	0.157	0.328	0.693
LU of J_0	0.060	0.124	0.255	0.524
LU of Φ	0.063	0.125	0.260	0.558
ILU(12) of J_i	0.090	0.190	0.393	0.842
ILU(12) of J_0	0.082	0.176	0.290	0.596
ILU(12) of Φ	0.076	0.152	0.318	0.718

VII. NUMERICAL EXPERIMENTS

In this section, we present the results of our numerical experiments. All tests are performed on a single core of a machine with Intel Core2 Duo 3-GHz CPU and 16-Gb memory, running a Slackware 13 64-bit Linux distribution. The problems are solved from a flat start, up to an accuracy of 10^{-6} p.u.

The test cases used are created using the UCTE winter 2008 model, as described in Section I. Table II shows the number of busses and lines in the test problems, as well as the number of non-zeros in the Jacobian matrix $\mathrm{nnz}(J)$. The naming convention used is uctewXXX, where XXX is the number of times the model is copied and interconnected.

The test problems are solved with inexact Newton-GMRES, using the Eisenstat and Walker forcing term strategy. We also report on the use of Bi-CGSTAB instead of GMRES.

Table III shows all the preconditioners reported on in this section. Note that ILU factorizations with less than 4 levels resulted in too slow GMRES convergence to be useful, and ILU with more than 12 levels took more time to calculate while yielding no significant improvement in the preconditioner.

A. Small Test Cases

In Table IV, the solution times for the smaller test systems are shown in seconds. For these systems, using the LU factorization of the initial Jacobian J_0 as a preconditioner is clearly the fastest. Preconditioners based on ILU(4) and ILU(8) were slightly slower than those based on ILU(12).

When solving a single power flow problem, all these methods are acceptable. When using the power flow solver as part of a larger solver that has to solve many power flow problems, as for example in contingency analysis, using the LU factorization of J_0 can lead to a significant benefit.

B. Scalability

First, preconditioners based on the Jacobian matrix are considered, i.e., $Q_i = J_i$. Fig. 1 shows the solution time in seconds versus the number of busses. Note that using a full LU factorization in this case is effectively the same as using a direct solver.

It is clear that the direct solver does not scale well in the problem size. The performance is acceptable up to a problem size of about 150 k busses, but deteriorates rapidly for larger problems. The largest problem, uctew256, took over an hour to

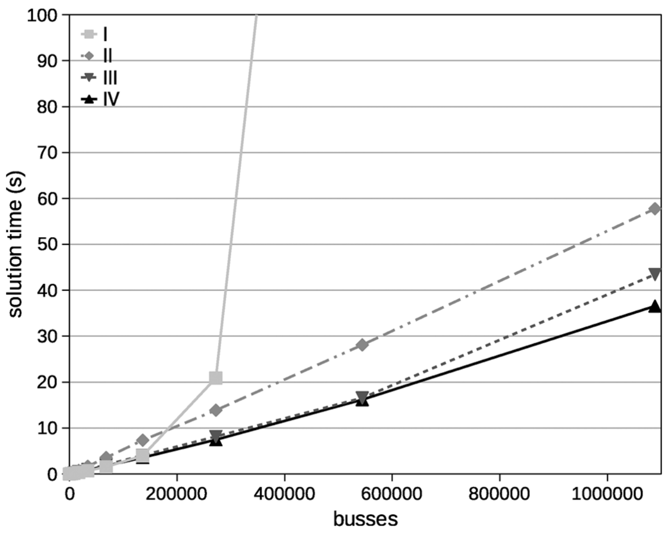


Fig. 1. Solution time in seconds with Jacobian preconditioner. I: LU of J_i, II: ILU(4) of J_i, III: ILU(8) of J_i, IV: ILU(12) of J_i.

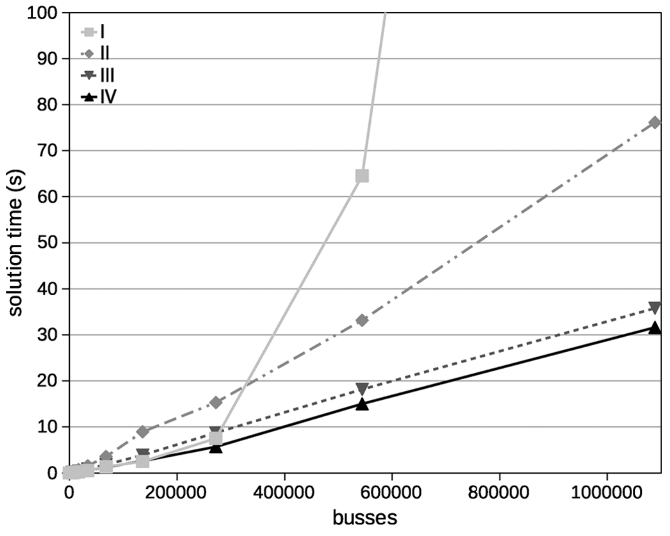


Fig. 2. Solution time in seconds with initial Jacobian preconditioner. I: LU of J_0, II: ILU(4) of J_0, III: ILU(8) of J_0, IV: ILU(12) of J_0.

solve with a direct linear solver, whereas the ILU(4), ILU(8), or ILU(12) factorization of J_i as preconditioner all solved the problem in less than a minute.

Next, preconditioners based on the initial Jacobian matrix are tested, i.e., $Q_i = J_0$. Fig. 2 shows the solution times for these preconditioners.

Since the preconditioner is the same in each iteration, only a single factorization has to be made. As the factorization is by far the most expensive computation, this is much faster than with a direct solver, where a new factorization has to be made in each iteration.

After the first iteration, J_0 will not be as good a preconditioner as J_i; thus, more GMRES iterations will be needed to converge. The results show that using ILU(4) of J_0 is slower than using ILU(4) of J_i. This is because the ILU(4) factorization is very fast, and most of the time is spend on the GMRES iterations. With J_0 as preconditioner, the number of GMRES iteration goes up by more than is saved by having to do only

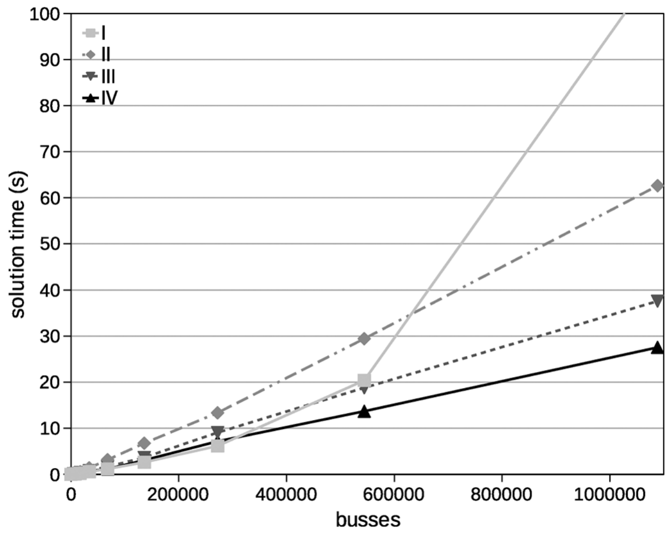


Fig. 3. Solution time in seconds with FDLF preconditioner. I: LU of Φ, II: ILU(4) of Φ, III: ILU(8) of Φ, IV: ILU(12) of Φ.

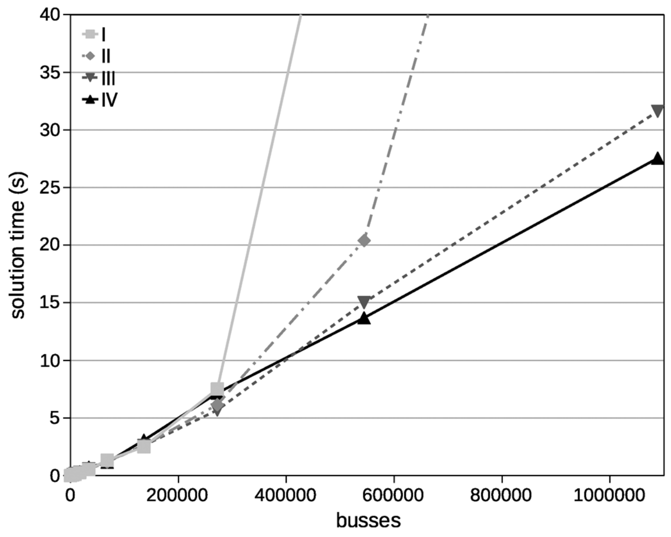


Fig. 4. Solution time for all test problems with the best performing methods. I: LU of J_0, II: LU of Φ, III: ILU(12) of J_0, IV: ILU(12) of Φ.

one factorization. However, for higher k-levels like ILU(12), the extra GMRES iterations needed when using J_0 cost less time than the extra factorizations needed for J_i.

Finally, the FDLF matrix is used as basis for the preconditioners, i.e., $Q_i = \Phi$. Fig. 3 shows the results.

Again only a single factorization has to be made, and the target matrix now has about half the non-zeros of the Jacobian. Because of this, using the LU factorization can stay competitive with the $\mathrm{ILU}(k)$ alternatives longer, up to about 300 k busses. However, for larger problems, the LU factorization scales badly again, and the ILU(12) factorization is a clear winner.

Inspection of the solution times showed that for our test problems, using preconditioners based on J_i (including direct solvers) was never better than using $Q_i = J_0$ or $Q_i = \Phi$. Furthermore, the ILU(12) factorization performed systematically better than the other ILU options. Therefore, in Fig. 4 the LU and ILU(12) factorizations of J_0 and Φ are compared.

Obviously, for the largest problems, the ILU(12) factorizations perform the best. The variant using ILU(12) of Φ is slightly faster, but close inspection of the solver output shows this is actually due to the inexact Newton-Krylov solver needing 8 Newton-Raphson iterations when using J_0, against 6 iterations for Φ. This has to do with being a bit lucky, or unlucky, at how the Newton step exactly turns out for a certain problem under the forcing term condition (6), much more than it has to do with the quality of the preconditioner.

C. GMRES versus Bi-CGSTAB

To validate the choice of GMRES as iterative solver for our power flow problems, Bi-CGSTAB was also tested with the same preconditioners. Bi-CGSTAB should be expected to outperform GMRES when a lot of GMRES iterations are needed, meaning that Bi-CGSTAB will become a better alternative when the preconditioner becomes worse.

The tests showed that for the largest test problem, Bi-CGSTAB was faster than GMRES when using ILU(4) preconditioners, and slightly slower than GMRES with ILU(12). Using ILU(4) with Bi-CGSTAB was still significantly slower than using ILU(12) with either iterative linear solver. The results for ILU(8) were hard to compare because an extra Newton iteration was needed when using Bi-CGSTAB.

D. Robustness

The Newton-Raphson method with exact linear solver is known to be locally convergent, meaning that it always converges to a solution provided that the starting point is close enough to that solution. The method can be made globally convergent using either line search or trust regions. For details, see for example [11]. Inexact Newton methods have also been shown to be locally convergent [18], and can again be made globally convergent with line search or trust regions.

Convergence of exact and inexact Newton methods are generally very close. Therefore, to compare the robustness of the methods, the linear solvers should be compared. As direct linear solvers are very robust, the focus of attention should be on the iterative linear solver.

The convergence of Krylov solvers heavily depends on the condition number of the coefficient matrix of the linear system to solve. This is exactly why preconditioning is such an important part of the solution process. The Jacobian systems of our test cases are in fact very ill-conditioned. For example, the condition number of the initial Jacobian J_0 of the uctew001 test case is 1.2×10^9. This also explains why lower quality preconditioners, like ILU(0), do not lead to convergence. However, with the preconditioners P_i based on LU and ILU(12) factorizations suggested in this paper, the condition numbers of the preconditioned coefficient matrices $J_i P_i^{-1}$ drop below 10, leading to very fast convergence.

To test the robustness of the methods suggested in this paper experimentally in the context of the power flow problem, we have run experiments on the uctew032 test case under different loading levels. Both the Newton-Raphson method with direct solver and the inexact Newton methods were able to solve the problem up to a loading level of 160%, but failed to converge

TABLE V
CONVERGENCE AT DIFFERENT LOADING LEVELS

load	N	GMRES iterations	$\tilde{\sigma}$
100%	6	21 [1,3,2,3,5,7]	3.5
110%	6	21 [1,3,2,3,5,7]	3.5
120%	6	22 [1,3,2,3,5,8]	4.1
130%	7	35 [1,3,2,3,6,7,13]	4.6
140%	7	37 [1,3,2,4,5,8,14]	5.4
150%	7	35 [1,3,2,4,6,7,12]	6.8
160%	7	34 [1,3,2,4,6,5,13]	10.4

at 170% without the help of line search or trust region techniques. It should be noted here that the solution of the power flow problem at the highest loading levels had such large voltage angles not to be of practical value, indicating that the solvers are well up for any practical loading levels of the power system.

Table V shows test results at different loading levels for the uctew032 test problem, using the LU factorization of J_0 as preconditioner. Presented are the number of Newton iterations N, the GMRES iterations (total, and the number performed in each Newton iteration), and an estimate $\tilde{\sigma}$ of the condition number of the preconditioned coefficient matrix in the last Newton iteration.

The solution of the problem with higher loading levels lies further away from the flat start than with lower load. Since the preconditioner is based on the Jacobian at a flat start, the Jacobian near the solution also differs more from the preconditioner for high loading levels. As a result, the condition number of the preconditioned coefficient matrix in the last Newton iteration goes up with the loading level. However, overall the condition number stays very small and GMRES convergence does not deteriorate visibly. It does take longer to solve the systems with high load, but this is due to Newton convergence suffering, and not the linear solver. Similar results hold for the other preconditioners suggested in this paper.

VIII. CONCLUSIONS

The experiments in this paper clearly illustrate the bad scaling of the LU factorization in the problem size. Because of this bad scaling, direct sparse solvers, but also Krylov methods with preconditioners based on the LU factorization, are not viable options for the solution of the linear systems that arise when solving very large power flow problems with the Newton-Raphson method. For the same reason, the classical implementation of the Fast Decoupled Load Flow method is also not viable for very large problems.

To solve very large power flow problems, a scalable solver to solve the linear systems is needed. In this paper, we have proposed to use an iterative linear solver, in particular GMRES, with an $\mathrm{ILU}(k)$ factorization of the Jacobian matrix J_i, the initial Jacobian matrix J_0, or the FDLF matrix Φ, as preconditioner.

We have shown that a good reordering strategy is essential to reduce the fill-in of the LU factorization, but also very important to improve the quality of ILU factorizations. In [6], we already showed the importance of choosing good forcing terms for the inexact Newton-Krylov method. Using the AMD reordering, and the Eisenstat and Walker forcing terms, we have conducted experiments on power flow problems up to a million busses.

The experiments show that the $\mathrm{ILU}(k)$ preconditioners scale very well for the tested problem sizes. Factorizing a preconditioner once at the start, i.e., using J_0 or Φ, generally performed better than refactorizing in each iteration, i.e., using J_i. For large problems, the best results were obtained with an ILU(12) factorization of either J_0 or Φ as a preconditioner, solving the largest test case with over a million busses in around 30 s.

For smaller problems, using the LU factorization of J_0 as a preconditioner for GMRES performed best, beating the direct solver by a substantial margin.

Further, the extra variables of the proposed methods lead to more flexibility when solving derived problems, like contingency analysis.

Experiments with Bi-CGSTAB showed to be very competitive for large problems. When using lower quality preconditioners, i.e., ILU factorization with lower k-levels, Bi-CGSTAB led to a faster power flow solver than GMRES with the same preconditioners. However, the best results were still those of GMRES with ILU(12)-based preconditioners.

Finally, experiments with a test case at different loading levels indicated that the proposed methods are equally robust as Newton-Raphson with a direct solver.

ACKNOWLEDGMENT

The authors would like to thank R. A. M. van Amerongen for his many contributions, providing hands-on experience, and invaluable insight on the subject of power flow analysis. The authors also would like to thank B. F. Smith for his help with the PETSc package, and UCTE/ENTSO-E for providing the UCTE study model, and their assistance on using it. Thanks are also extended to the reviewers for their many insightful comments.

REFERENCES

[1] W. F. Tinney and C. E. Hart, “Power flow solution by Newton’s method,” *IEEE Trans. Power App. Syst.*, vol. PAS-86, no. 11, pp. 1449–1449, Nov. 1967.

[2] W. F. Tinney and J. W. Walker, “Direct solutions of sparse network equations by optimally ordered triangular factorization,” *Proc. IEEE*, vol. 55, no. 11, pp. 1801–1809, Nov. 1967.

[3] A. J. Flueck and H. D. Chiang, “Solving the nonlinear power flow equations with an inexact Newton method using GMRES,” *IEEE Trans. Power Syst.*, vol. 13, no. 2, pp. 267–273, May 1998.

[4] F. de León and A. Semlyen, “Iterative solvers in the Newton power flow problem: Preconditioners, inexact solutions and partial Jacobian updates,” *Proc. Inst. Elect. Eng., Gen., Transm., Distrib*, vol. 149, no. 4, pp. 479–484, Jul. 2002.

[5] D. Chaniotis and M. A. Pai, “A new preconditioning technique for the GMRES algorithm in power flow and $P-V$ curve calculations,” *Elect. Power Energy Syst.*, vol. 25, pp. 239–245, 2003.

[6] R. Idema, D. J. P. Lahaye, C. Vuik, and L. van der Sluis, “Fast Newton load flow,” in *Proc. 2010 IEEE PES Transmission and Distribution Conf. Expo.*, Apr. 2010, pp. 1–7.

[7] Y.-S. Zhang and H.-D. Chiang, “Fast Newton-FGMRES solver for large-scale power flow study,” *IEEE Trans. Power Syst.*, vol. 25, no. 2, pp. 769–776, May 2010.

[8] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, B. F. Smith, and H. Zhang, PETSc Users Manual, Argonne National Laboratory, Tech. Rep. ANL-95/11-Revision 3.1, 2010. [Online]. Available: http://www.mcs.anl.gov/petsc/.

[9] P. Schavemaker and L. van der Sluis, *Electrical Power System Essentials*. Chichester, U.K.: Wiley, 2008.

[10] R. Idema, D. J. P. Lahaye, and C. Vuik, Load Flow Literature Survey, Delft Institute of Applied Mathematics, Delft University of Technology, Report 09-04, 2009.

[11] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[12] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*. New York: Oxford Univ. Press, 1986.

[13] T. A. Davis, *Direct Methods for Sparse Linear Systems*. Philadelphia, PA: SIAM, 2006.

[14] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 856–869, 1986.

[15] H. A. van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for solution of nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 13, pp. 631–644, 1992.

[16] G. L. G. Sleijpen, H. A. van der Vorst, and D. R. Fokkema, " $BiCGstab(\ell)$ and other hybrid Bi-CG methods," *Numer. Alg.*, vol. 7, pp. 75–109, 1994.

[17] P. Sonneveld and M. B. van Gijzen, " $\mathrm{IDR}(s)$: A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations," *SIAM J. Sci. Comput.*, vol. 31, no. 2, pp. 1035–1062, 2008.

[18] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact Newton methods," *SIAM J. Numer. Anal.*, vol. 19, no. 2, pp. 400–408, 1982.

[19] S. C. Eisenstat and H. F. Walker, "Choosing the forcing terms in an inexact Newton method," *SIAM J. Sci. Comput.*, vol. 17, no. 1, pp. 16–32, 1996.

[20] R. S. Dembo and T. Steihaug, "Truncated-Newton algorithms for large-scale unconstrained optimization," *Math. Program.*, vol. 26, pp. 190–212, 1983.

[21] A. Hohmann, "Inexact Gauss Newton methods for parameter dependent nonlinear problems," Ph.D. dissertation, Freie Universität, Berlin, Germany, 1994.

[22] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA: SIAM, 2000.

[23] J. A. Meijerink and H. A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix," *Math. Computat.*, vol. 31, no. 137, pp. 148–162, Jan. 1977.

[24] J. A. Meijerink and H. A. van der Vorst, "Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems," *J. Computat. Phys.*, vol. 44, no. 1, pp. 134–155, 1981.

[25] B. Stott and O. Alsac, "Fast decoupled load flow," *IEEE Trans. Power App. Syst.*, vol. PAS-93, no. 3, pp. 859–869, 1974.

[26] R. A. M. van Amerongen, "A general-purpose version of the fast decoupled loadflow," *IEEE Trans. Power Syst.*, vol. 4, no. 2, pp. 760–770, May 1989.

[27] A. J. Monticelli, A. Garcia, and O. R. Saavedra, "Fast decoupled load flow: Hypothesis, derivations, and testing," *IEEE Trans. Power Syst.*, vol. 5, no. 4, pp. 1425–1431, Nov. 1990.

[28] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM J. Sci. Comput.*, vol. 14, no. 2, pp. 461–469, Mar. 1993.

[29] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*. New York: Academic, 2001.

[30] M. Yannakakis, "Computing the minimum fill-in is NP-complete," *SIAM J. Alg. Disc. Meth.*, vol. 2, no. 1, pp. 77–79, Mar. 1981.

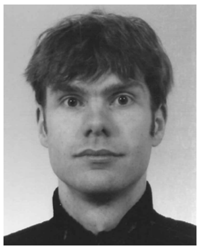
[31] P. R. Amestoy, T. A. Davis, and I. S. Duff, "An approximate minimum degree ordering algorithm," *SIAM J. Matrix Anal. Appl.*, vol. 17, no. 4, pp. 886–905, Oct. 1996.

[32] T. A. Davis, "A column pre-ordering strategy for the unsymmetric-pattern multifrontal method," *ACM Trans. Math. Softw.*, vol. 30, no. 2, pp. 165–195, Jun. 2004.

[33] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, "A supernodal approach to sparse partial pivoting," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 3, pp. 720–755, 1999.

[34] X. S. Li and J. W. Demmel, "SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems," *ACM Trans. Math. Softw.*, vol. 29, no. 2, pp. 110–140, Jun. 2003.

[35] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM J. Matrix Anal. Appl.*, vol. 23, no. 1, pp. 15–41, 2001.



Reijer Idema (S'10) received the M.Sc. degree in applied mathematics (computational science and engineering specialization) at the Delft University of Technology, Delft, The Netherlands, in 2007. Currently, he is pursuing the Ph.D. degree at the Numerical Analysis research group of the Delft Institute of Applied Mathematics, Delft University of Technology.

Domenico J. P. Lahaye received the M.Sc. degree in applied mathematics at the Free University of Brussels, Brussels, Belgium, in 1994, the postgraduate degree in mathematics for the industry at the Eindhoven University of Technology, Eindhoven, The Netherlands, in 1996, and the Ph.D. degree in computer science at the Catholic University of Leuven, Leuven, Belgium, in 2001.

After having held positions at the Center for Advanced Studies, Research and Development in Sardinia, and at the National Research Center for Mathematics and Computer Science in the Netherlands, he joined the Numerical Analysis research group of the Delft Institute of Applied Mathematics, Delft University of Technology, Delft, The Netherlands, as an Assistant Professor in 2007.

Cornelis Vuik received the M.Sc. degree in applied mathematics at the Delft University of Technology, Delft, The Netherlands, in 1982. After a short stay at the Philips Research Laboratories, he received the Ph.D. degree in mathematics at Utrecht University, Utrecht, The Netherlands, in 1988.

Thereafter, he became employed at the Delft University of Technology, where he is full Professor of the Numerical Analysis research group. In 2007, he additionally became Director of the Delft Center of Computational Science and Engineering.

Lou van der Sluis (SM'86) received the M.Sc. degree in electrical engineering at the Delft University of Technology, Delft, The Netherlands, in 1974.

He joined the KEMA High Power Laboratory in 1977. In 1990, he became a part-time Professor at the Delft University of Technology, and since 1992, he has been employed as a full Professor of the Power Systems Department of the Delft University of Technology.