Fast Newton Load Flow

Reijer Idema, Domenico Lahaye, Kees Vuik, and Lou van der Sluis, *Senior Member, IEEE*

***Abstract*—The Newton-Raphson method is widely used to solve load flow problems. Traditionally a direct solver is used to solve the linear systems within this method. In this paper we explore the use of an iterative method to solve the linear systems, leading to an inexact Newton-Krylov method. The main parameters of this method are the preconditioner and the forcing terms. Several candidate choices for these parameters are discussed and tested. With the proper preconditioner, and forcing terms, the inexact Newton-Krylov method is shown to greatly improve on using a direct solver.**

***Index Terms*—load flow analysis, Newton-Raphson method**

I. Introduction

THE load flow, or power 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.

Load flow computations bring insight in the steady-state behavior of a power system. This is needed in many control and planning applications. For example, whenever power system components have to be taken out of service, say for maintenance purposes, it is crucial to know whether the power system will still function within system limits, or that additional measures have to be taken. Moreover, a power system has to be at least $n-1$ secure. This means that if any one component fails, the power system still functions properly. These are typical problems that can be solved by doing load flow computations on the power system network.

The consumption of electricity keeps on rising every year. As a result, power systems grow larger and larger to supply all consumers with their needs. And with the privatization of the electricity market, power systems are operated closer and closer to their limits for economic reasons. More and more nation wide 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.

With ever-growing problem sizes, more and more load flow computations will have to be run. For a 1000 component system, a naive full $n-1$ security analysis would require 1000 runs of a 1000 component load flow problem. But for a 10000 component system the same analysis already requires 10 times that amount of runs, on a 10 times bigger problem, meaning a hundred times the computational work. Furthermore, in a larger power system it is much more likely to have multiple failures at the same time. Therefore $n-2$ security analysis is already being done regularly, taking huge amounts of computational effort.

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

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

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 usually decentralized, i.e., connected to the distribution network at consumer level. Also, renewable sources are mostly natural and often have an uncontrollable power output.

The incorporation of renewable energy sources, with fluctuating power output, adds a stochastic component to the network computations. This can be dealt with by running a large amount of load flow computations in a Monte Carlo simulation. Traditionally load flow calculations are 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 load flow calculations to also incorporate the distribution network. This would result in load flow problems of sheer massive size.

Motivated by the above described developments, we explore mathematical techniques for load flow problems, that are particularly well equipped to deal with very large problem sizes. For more details on power systems we refer to [1].

Over the years, a lot of effort has been put into solving load flow problems efficiently. For a concise overview of earlier methods see [2]. In modern applications, the most widely used techniques are the Newton-Raphson method with a direct linear solver [3], [4], and the Fast Decoupled Load Flow (FDLF) method [5], [6], [7]. For an overview of the derivation of the load flow problem formulation used in this paper, and the application of the two above mentioned solution methods on this problem formulation, we also refer to [8].

The Newton-Raphson method is a very powerful tool. There are multiple techniques available to assure global convergence, while local convergence is quadratic, see for example [9]. The FDLF method is a very fast load flow method, but lacks some of the convergence properties of the Newton process. When applied to critical systems, or systems with strongly varying R/X ratios, the FDLF method may well fail to converge [7].

In this paper we focus on the Newton-Raphson method. But where traditionally a direct linear solver is used in each Newton iteration, we use iterative linear solution methods. Similar work was done in [10], [11], [12].

Iterative linear solvers, also referred to as Krylov methods, are generally a lot faster than direct solvers for large sparse problems. For an overview of iterative solvers for sparse linear systems see [13]. We show that for larger load flow problems a Newton method with an iterative linear solver, also called a Newton-Krylov method, offers a huge improvement over using a direct solver, and that even for small load flow problems it can outperform Newton with a direct solver.

In Section II we define the linear system that has to be solved in each Newton iteration. Then in Section III we motivate a choice of iterative linear solver and preconditioner, while in Section IV we discuss the accuracy to which to solve the linear system in each iteration. In Section V we consolidate our work into numerical experiments, and compare the results with the traditional methods. Finally, in Section VI we present our conclusions.

Numerical experiments are performed in MATLAB, using MATPOWER[1] complemented with our own code. Our main case is a power system network that is derived from the UCTE[2] winter 2008 study model, and consists of 4253 busses and 7191 lines. All load flow solves are done with a flat start, and up to an accuracy of 10^{-8}.

II. PROBLEM FORMULATION

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

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

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

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 [1], [8].

We define the power mismatch function as

$$\boldsymbol{F}(\boldsymbol{x}) = \left[\begin{array}{c} P_i - \sum_{k=1}^{N} |V_i|\,|V_k|\,(G_{ik}\cos\delta_{ik} + B_{ik}\sin\delta_{ik}) \\ Q_i - \sum_{k=1}^{N} |V_i|\,|V_k|\,(G_{ik}\sin\delta_{ik} - B_{ik}\cos\delta_{ik}) \end{array} \right] \qquad (3)$$

where $\boldsymbol{x}$ is the vector of voltage angles and magnitudes. Then we can reformulate the load flow problem (1), (2), as finding $\boldsymbol{x}$ such that

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

The Newton-Raphson method is an iterative method that, given an estimate solution $\boldsymbol{x}_i$, calculates a new estimate $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{s}_i$, where the update $\boldsymbol{s}_i$ is an approximation of the vector $\tilde{\boldsymbol{s}}_i$ for which

$$\boldsymbol{F}(\boldsymbol{x}_i + \tilde{\boldsymbol{s}}_i) = \boldsymbol{0}. \qquad (5)$$

This update $\boldsymbol{s}_i$ is calculated by solving the first order Taylor approximation of equation (5), i.e.,

$$J_i \boldsymbol{s}_i = -\boldsymbol{F}_i, \qquad (6)$$

where $\boldsymbol{F}_i = \boldsymbol{F}(\boldsymbol{x}_i)$, and $J_i = J(\boldsymbol{x}_i)$ is the Jacobian matrix of $\boldsymbol{F}$ in $\boldsymbol{x}_i$. For details see for example [9].

[1]See http://www.pserc.cornell.edu/matpower/

[2]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/

III. PRECONDITIONED GMRES

The Generalized Minimal Residual method (GMRES) [14] is probably the most widely used iterative solver for general linear systems. It is a minimal residual method, meaning that in each iteration the residual error is minimized within the Krylov subspace. In other words, in each iteration the solution is as good as it can get with a Krylov method.

The drawback of GMRES is that it does not have short recurrences. In each iteration the amount of vectors that has to be stored in memory increases, as does the number of vector operations that has to be performed. As a result, GMRES becomes very slow when a lot of iterations are needed.

Preconditioning is an essential tool to bring down the number of iterations needed by iterative linear solvers. Instead of the original linear system (6), a preconditioned system

$$M_i^{-1} J_i \boldsymbol{s}_i = -M_i^{-1} \boldsymbol{F}_i \qquad (7)$$

is solved, where M_i is called the preconditioner. The idea is to choose M_i such that it is easy to solve systems of the form $M_i \boldsymbol{y} = \boldsymbol{b}$, while M_i resembles J_i in such a way that $M_i^{-1} J_i$ is well-conditioned. For details on preconditioning techniques see again [13].

The efficiency of any iterative method generally relies heavily on a good preconditioner. The preconditioners that we consider are based on the Jacobian itself, and on the FDLF matrix

$$\Phi = \left[\begin{array}{cc} B' & 0 \\ 0 & B'' \end{array} \right], \qquad (8)$$

according to the BX scheme advocated in [6]. Note that we did also try other FDLF schemes, but they yield worse results. The FDLF matrix can be seen as an approximate Schur complement of the initial Jacobian matrix [7]. As such, Φ can be expected to be a good preconditioner, with only half the non-zeros of the Jacobian matrix. The use of the FDLF matrix as preconditioner was also noted in [10].

As mentioned, we need to be able to solve systems of the form $M\boldsymbol{y} = \boldsymbol{b}$ efficiently. To this end we factorize the preconditioner, and pass the factors to the GMRES method. The factorization used is the LU decomposition, a factorization into a lower triangular matrix L, and an upper triangular matrix U, see for example [15]. We also tried the incomplete LU (ILU) decomposition, with varying drop tolerances. However we found that, for currently available problem sizes, the gain in sparsity with the ILU factorization did not nearly outweigh the loss in quality of the preconditioner. Note that for very large problems, say over 100,000 busses depending on the hardware, doing a complete LU factorization may become problematic. In this case ILU factorizations offer a useful alternative.

To test the preconditioners, we solve the initial Jacobian system $J_0 \boldsymbol{s}_0 = -\boldsymbol{F}_0$ for the UCTE test network described at the end of Section I. The system is solved with preconditioned GMRES, up to an accuracy of 10^{-8}. Table I shows the number of GMRES iterations, the time used by GMRES, and the time spent on the factorization, for the suggested preconditioners.

It appears that the Jacobian is the ideal preconditioner, as it converges in one iteration. However, if we choose J_i as a preconditioner in each iteration i, we are actually solving the

TABLE I
PRECONDITIONER FACTORIZATION TIME AND GMRES CONVERGENCE

preconditioner	GMRES (it)	GMRES (s)	factorization (s)
none	1886	388	0
LU(J_0)	1	0.004	0.083
LU(Φ)	19	0.072	0.058

linear systems with a direct solver. All the computational effort is diverted from the GMRES method to the factorizations. Instead we will choose the preconditioner at the start, factorize it once, and use these factors throughout all Newton iterations. In other words, in each Newton iteration we solve

$$M^{-1}J_i\boldsymbol{s}_i = -M^{-1}\boldsymbol{F}_i, \tag{9}$$

where M is a preconditioner chosen and factored at the start.

In load flow computations, the initial iterate $\boldsymbol{x}_0$ is generally reasonably close to the solution, even at a flat start. Thus the iterates $\boldsymbol{x}_i$ will be relatively close together, and the Jacobian matrices J_i should not vary too much. As such, J_0 is expected to be a good preconditioner for all J_i. Obviously, for the FDLF preconditioner Φ we had no other choice to begin with than to keep it constant through all Newton iterations, as there is only one such matrix available.

Using this strategy, the factorization becomes a sort of preprocessing step outside of the Newton iteration loop, while the factors serve to reduce the computational time of each Newton iteration. The price one pays is in the quality of the preconditioner. Where J_0 is in a way the best possible preconditioner for the first iteration, it will not be as good for the Jacobian system of later iterations.

Table II shows the number of GMRES iterations needed in each Newton iteration for both preconditioners. Each GMRES solve is again done up to an accuracy of 10^{-8}. This nicely illustrates that both J_0 and Φ remain very good preconditioners throughout the Newton process. The FDLF preconditioner Φ requires almost double the GMRES iterations total, compared to J_0. However, recall that it also has about half the non-zeros, and thus each GMRES iteration will be faster. Both preconditioners will therefore be considered in our numerical experiments in Section V.

TABLE II
PRECONDITIONED GMRES ITERATIONS

preconditioner	1	2	3	4	5	6	total
LU(J_0)	1	16	11	10	11	10	59
LU(Φ)	19	25	19	18	17	18	116

Methods like Bi-CGSTAB [16], and IDR(s) [17] use short recurrences at the loss of the minimal residual property. In other words, they only need constant work per iteration, where GMRES needs more work in each subsequent iteration, but they generally need more iterations to converge. Note that Bi-CGSTAB was proposed as method of choice in [11].

With the low GMRES iteration count needed with the proposed preconditioners, the long recurrences of GMRES do not pose a problem, especially after application of the forcing term strategies that we present in Section IV. We can use GMRES without restarting it, as was also concluded in [12]. As such there is no reason to give up the minimal residual property of GMRES, for a short recurrence method.

To further illustrate the quality of the preconditioners, Fig. 1–3 show the eigenvalues of the coefficient matrix of the linear system in the final Newton iteration, without preconditioning, with the initial Jacobian preconditioner, and with the FDLF preconditioner, respectively.

Without preconditioner there are a lot of eigenvalues close to 0, as well as very large eigenvalues with real and imaginary parts of order 10^4. This makes the system hard to solve for Krylov methods. Both preconditioners J_0, and Φ, manage to cluster the eigenvalues nicely around 1, resulting in well-conditioned coefficient matrices.

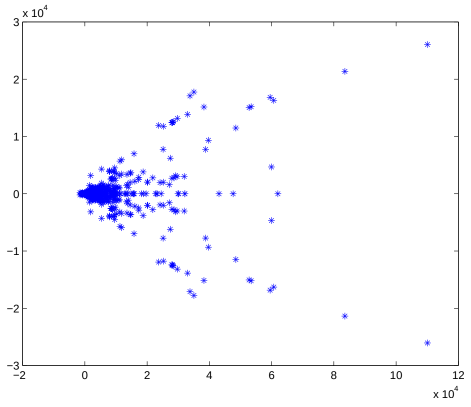

Fig. 1. Spectrum of J_6 (no preconditioning)

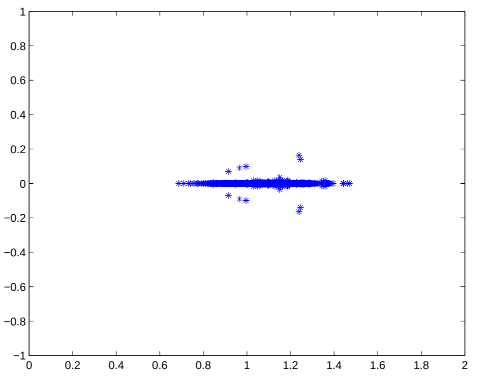

Fig. 2. Spectrum of $M^{-1}J_6$, with $M = J_0$

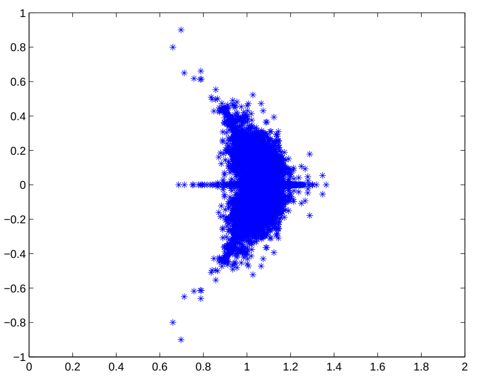

Fig. 3. Spectrum of $M^{-1}J_6$, with $M = \Phi$

IV. Inexact Newton Method

A direct solver always solves the linear system up to machine precision. By contrast, an iterative solver calculates a solution that gets better in each iteration. With an iterative method, therefore, a rough approximation of the solution takes less time to calculate than a very good approximation. This fact inspires the use of an inexact Newton method.

An inexact Newton method, is a Newton process where in each iteration the Jacobian system (6) is solved up to a precision

$$\|J_i \boldsymbol{s}_i + \boldsymbol{F}_i\| \leq \eta_i \|\boldsymbol{F}_i\|, \tag{10}$$

where η_i is called the forcing term. As long as the forcing terms are chosen such that $\eta_i \to 0$ for $i \to \infty$, the inexact Newton method has superlinear convergence near the solution [18].

The idea is that the Newton step $\boldsymbol{s}_i$ is in itself an approximation, given by the linearization of the power mismatch function $\boldsymbol{F}$ around the current iterate $\boldsymbol{x}_i$. In other words, let $\hat{\boldsymbol{s}}_i$ be the ideal step, such that $\hat{\boldsymbol{x}} = \boldsymbol{x}_i + \hat{\boldsymbol{s}}_i$ is the exact solution to the load flow problem, then there is some step error $\|\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i\|$. Note that this step error is equal to the error in the solution after taking the step $\boldsymbol{s}_i$, because

$$\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i = (\boldsymbol{x}_i + \boldsymbol{s}_i) - (\boldsymbol{x}_i + \hat{\boldsymbol{s}}_i) = \boldsymbol{x}_{i+1} - \hat{\boldsymbol{x}}. \tag{11}$$

Solving system (6) to such a high accuracy that the error in $\boldsymbol{s}_i$ is much smaller than the step error $\|\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i\|$ is thus a waste of computational time, since it will not significantly reduce the error in $\boldsymbol{x}_{i+1}$. As the Newton-Raphson method converges quadratically near the solution, we can expect the step error to reduce faster in later iterations. We can start with a relative large forcing term η_i, without losing the fast convergence of the Newton process, as long as we sufficiently reduce η_i in each iteration

Fig. 4 shows a typical convergence plot of GMRES, preconditioned with the initial Jacobian, in one of the later Newton iterations. The relative residual error that is set out on the y-axis is $\frac{\|J_i \boldsymbol{s}_i + \boldsymbol{F}_i\|}{\|\boldsymbol{F}_i\|}$. This is exactly the target value to get below the forcing term η_i in equation (10).

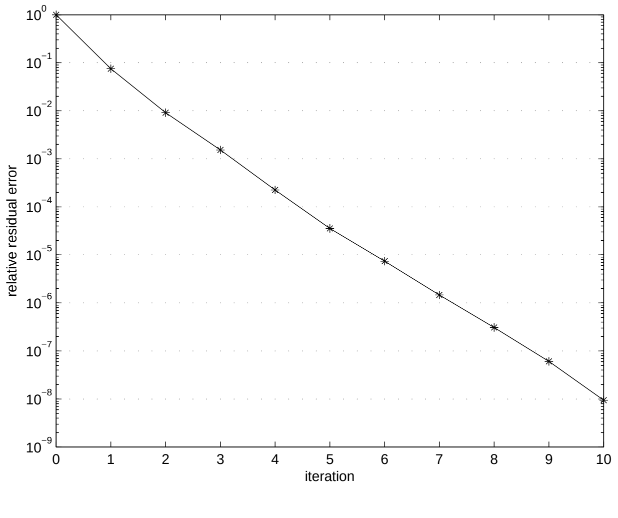

Fig. 4. Convergence preconditioned GMRES

The convergence is approximately linear. It only takes 1 iteration to satisfy equation (10) for $\eta_i = 10^{-1}$, 2 iterations for $\eta_i = 10^{-2}$, and 5 iterations for $\eta_i = 10^{-4}$. To solve up an accuracy of 10^{-8} already takes 10 iterations. This nicely illustrates the computational time that can be saved by solving to a lower accuracy. Especially because GMRES uses more time and memory per iteration in later iterations.

Note that GMRES generally converges superlinearly [19]. However, in our case the eigenvalues of the preconditioned matrix are clustered tightly around 1, as shown in Fig. 2. This leads to very fast, albeit approximately linear, convergence.

From Table II we know that for $\eta_i = 10^{-8}$, a total of 59 GMRES iterations is needed, over 6 Newton iterations. If we set η_i to minimize the amount of GMRES iterations in each Newton iteration, while maintaining the same Newton convergence as full accuracy solves, only 16 GMRES iterations are needed, over the same 6 Newton iterations. However, in practice it is generally not possible to know such ideal tolerance settings without solving the complete problem first.

Through the years, a lot of effort has been invested in finding good strategies to choose the forcing terms in a Newton process. Here we consider a few of these strategies. The application of these strategies on our test problem, with the J_0 preconditioner, are shown in Table III.

TABLE III
GMRES iterations - Forcing term strategies

	1	2	3	4	5	6	7	8	9	10	total
$\eta_i = 10^{-8}$	1	16	11	10	11	10					59
$\eta_i = 10^{-1}$	1	2	2	2	2	2	2	2	2	2	19
strategy 1	1	2	2	2	2	3	6				18
strategy 2	1	3	1	3	3	6	7				24
strategy 3	1	2	2	3	5	9					22
ideal	1	1	1	2	4	7					16

A very simple strategy is to use constant low accuracy solves. Such a strategy can lead to a great reduction in the number of total GMRES iterations. However this generally comes at the cost of doing quite a few more Newton iterations, as the superlinear convergence of the Newton process is lost. For example, choosing $\eta_i = 10^{-1}$, as was used in [10], leads to 10 Newton iterations. As the overhead on extra Newton iterations is quite high, we do not further consider such a strategy.

Strategy 1 is to choose

$$\eta_i = \min\left\{\frac{1}{2^i}, \|\boldsymbol{F}_i\|\right\}. \tag{12}$$

This allows for superlinear convergence when the power mismatch is still large, while switching to quadratic convergence when nearing the solution. A similar strategy was used in [20].

Strategy 2, as proposed in [21], sets for $i > 0$

$$\eta_i = \left| \frac{\|\boldsymbol{F}_i\| - \|\boldsymbol{F}_{i-1} + J_{i-1}\boldsymbol{s}_{i-1}\|}{\|\boldsymbol{F}_{i-1}\|} \right|, \tag{13}$$

while safeguarding from oversolving by adding the rule

$$\text{if } \eta_{i-1}^{(1+\sqrt{5})/2} > \frac{1}{10}, \text{ then } \eta_i = \max\left\{\eta_i, \eta_{i-1}^{(1+\sqrt{5})/2}\right\}. \tag{14}$$

To start the process we use $\eta_0 = 0.1$.

Strategy 3, finally, is to choose

$$\eta_i = \frac{\varepsilon_i}{1+\varepsilon_i}, \tag{15}$$

where

$$\varepsilon_i = \frac{\beta}{2}\min\{1, h_i\}, \tag{16}$$

with

$$h_i = \begin{cases} \frac{2-\beta}{1+\beta} & i = 0, \\ \frac{1+\beta}{2(1-\varepsilon_{i-1})}h_{i-1}^2 & i > 0. \end{cases} \tag{17}$$

This is the affine contravariant strategy derived in [22]. It was also applied to load flow problems in [11]. For our experiments we used $\beta = 1$. Note that the forcing terms of strategy 3, unlike the other strategies, do not depend on the problem.

From Table III it is clear that strategies 1–3 perform very well. For this particular problem, strategy 1 reduces the GMRES iterations the most, though at the cost of an extra Newton iteration. Strategy 3 sets the forcing terms a little sharper, resulting in more GMRES iterations, but no extra Newton iteration.

Which strategy is the best will generally depend on the problem. Larger forcing terms will result in less GMRES iterations per Newton iteration, but may result in extra Newton iterations. The perfect balance depends not only on the problem, but also on the computational cost of extra Newton iterations in the used solver implementation.

V. Numerical Experiments

The theory and results of the previous sections can be summarized as Algorithm 1. The main ideas are the choice of the preconditioner, and the complete LU factorization of this preconditioner, at the start, and the solution of the Jacobian system by means of preconditioned GMRES, up to a variable tolerance η_i, in each Newton iteration.

Algorithm 1 Inexact Newton-Krylov Load Flow solver

```
 1: $i = 0$
 2: choose initial solution $\boldsymbol{x}_0$
 3: calculate power mismatch $\boldsymbol{F}_0$
 4: choose preconditioner $M \in \{J_0, \Phi\}$
 5: factorize preconditioner $LU = M$
 6: while not converged do
 7:     calculate Jacobian $J_i$
 8:     solve $J_i\boldsymbol{s}_i = -\boldsymbol{F}_i$ with $LU$-preconditioned GMRES
        up to a relative residual error tolerance of $\eta_i$
 9:     update $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{s}_i$
10:     calculate power mismatch $\boldsymbol{F}_{i+1}$
11:     $i = i + 1$
12: end while
```

It is important to realize that the choice of the preconditioner has no influence on the number of Newton iterations needed to solve the problem, but only on the number of GMRES iterations. Likewise, the forcing term η_i only influences the number of GMRES iterations needed, as long as it is chosen small enough in each iteration. If it is chosen too big, extra Newton iterations will be needed. But even then it could still very well result in a gain in overall speed.

Table IV shows the results of running our test network on a desktop computer. The methods used are FDLF, Newton with a direct solver (N-D), and Newton-Krylov (N-K) using GMRES with different combinations of preconditioners and forcing term strategies. The forcing term strategies 1–3, are as described in the previous section. Strategy 0 means that the linear system was solved to an accuracy of 10^{-8} in each iteration.

Table columns 4 and 5 hold the number of iterations, and total time, needed by the nonlinear solver. Note that the iteration count of 31 for FDLF means that there were 16 P-iterations, and 15 Q-iterations. Column 6 shows the time spent on building, and factorizing, the preconditioner, while columns 7 and 8 hold the data on the linear solver. The final column shows the time spent on calculating Jacobian matrices. All times are measured in seconds.

TABLE IV
Test case 4253 busses, 7191 lines

			nonlinear		preco	linear		jacob
	preco	η	iter	time	time	iter	time	time
FDLF	n/a	n/a	31	0.35	n/a	n/a	n/a	n/a
N-D	n/a	n/a	6	0.84	n/a	n/a	0.50	0.18
N-K	J_0	0	6	0.64	0.08	59	0.21	0.18
	J_0	1	7	0.52	0.08	18	0.06	0.22
	J_0	2	7	0.55	0.08	24	0.08	0.22
	J_0	3	6	0.50	0.08	22	0.08	0.18
N-K	Φ	0	6	0.88	0.14	116	0.43	0.18
	Φ	1	7	0.61	0.14	38	0.10	0.22
	Φ	2	7	0.72	0.14	52	0.19	0.22
	Φ	3	6	0.64	0.14	46	0.17	0.18

When comparing the total solve times from column 5, we see that the Newton-Krylov methods are a nice improvement on Newton with a direct solver. The best results are offered by using the J_0 preconditioner, with forcing term strategies 1 and 3. To truly appreciate the gain, we examine the linear solve times in column 8. At the cost of 0.08s to build a preconditioner, the total linear solve time is reduced from 0.50s for direct solves, to a mere 0.06s when using forcing term strategy 1.

It is also interesting to note the effect of the stricter forcing terms of strategy 3, compared to strategy 1, for this particular load flow problem. For both preconditioners, strategy 3 needs one Newton iteration less than strategy 1, at the cost of some extra GMRES iterations. In the case of the J_0 preconditioner this leads to a slightly reduced solve time overall, whereas for the Φ preconditioner the extra GMRES iterations dominate, making strategy 3 slightly slower than strategy 1.

Although both preconditioners perform very well, using J_0 as a preconditioner for GMRES clearly outperforms using Φ. It is a better quality preconditioner, and leads to about half the number of iterations that are needed with Φ. Recall that Φ has only half the non-zero entries of J_0. This does speed up each GMRES iteration slightly, but not by such an amount that it rivals the performance of the J_0 preconditioner. Using an inexact Newton-Krylov method with the J_0 preconditioner, and properly chosen forcing terms, comes a lot closer to the speed of FDLF than Newton with a direct solver, while retaining the advantageous convergence properties of the Newton process.

Due to the local quadratic convergence of the Newton method, the iterates in the last few iterations are very close together. As such, the Jacobians of these iterations also hardly vary. This could inspire the idea to choose a new preconditioner J_i, and factorize it, halfway the Newton process. However, note that with the J_0 preconditioner, and a proper forcing term strategy, the total linear solve time is actually lower than that of a single factorization. An extra factorization can thus only lead to increased computational time overall.

Also note the computational time spent on calculating Jacobian matrices. For the J_0 preconditioned Newton-Krylov methods, with forcing term strategy 1–3, the cost of these Jacobians is higher than that of building the preconditioner, and doing all linear solves, together. We feel that this is in large part due to the MATLAB implementation, because element operations are slow in MATLAB, while the preferred matrix-vector operations are not ideal for this particular problem. An implementation in a low level language, such as C or Fortran, should reduce the cost of calculating the Jacobian matrices.

It is well-known that iterative linear solvers generally perform better for large sparse systems than direct solvers. The larger the power system network is, the larger the gain we can expect from Newton-Krylov methods, over Newton methods with a direct solver. To illustrate that the use of Newton-Krylov methods is not restricted to large systems, we also tested our solver on the IEEE 300 bus test case. The results are shown in Table V. Even for such a small network, proper preconditioning and forcing terms lead to reduced computational time.

TABLE V
TEST CASE IEEE 300

			nonlinear		preco	linear		jacob
	preco	η	iter	time	time	iter	time	time
FDLF	n/a	n/a	17	0.020	n/a	n/a	n/a	n/a
N-D	n/a	n/a	5	0.039	n/a	n/a	0.023	0.012
N-K	J_0	0	5	0.036	0.005	26	0.013	0.012
	J_0	1	5	0.031	0.005	12	0.008	0.012
	J_0	2	6	0.037	0.005	17	0.010	0.014
	J_0	3	6	0.037	0.005	18	0.011	0.014
N-K	Φ	0	5	0.073	0.010	89	0.044	0.012
	Φ	1	6	0.049	0.010	32	0.016	0.014
	Φ	2	5	0.043	0.010	29	0.014	0.012
	Φ	3	6	0.052	0.010	40	0.021	0.014

VI. CONCLUSIONS

We treated the inexact Newton-Krylov method to solve load flow problems. Preconditioned GMRES was motivated as the iterative linear solver of choice, with the complete LU factorization of the initial Jacobian J_0, or the FDLF matrix Φ, as preconditioner. The best results were attained with the J_0 preconditioner. Three strategies for choosing the forcing terms η_i were also introduced and tested. All performed very well, greatly decreasing the iterations, and time, needed by the linear solver. However, which one is the best depended on the problem, and the preconditioner used. From the presented numerical experiments, strategy 1 seems a solid choice overall, always being the best, or close to it, and never the worst.

The treated method offers a significant improvement on Newton load flow with a direct solver. With the proper choice of forcing terms, the convergence properties of the Newton method are preserved, while the time spent on linear solves is greatly reduced. The larger the power system network is, the greater the reduction is expected to be.

The proposed method comes a lot closer to FDLF than traditional Newton load flow, in terms of computational time. It provides a fast alternative in cases where FDLF convergence can not be guaranteed beforehand. Note that solving a load flow problem to low accuracy can favor FDLF, because the Newton method may not reach the stage of quadratic convergence. On the other hand, solving from a good starting solution favors the inexact Newton-Krylov method. Not only can quadratic convergence be reached early, also the J_0 preconditioner will remain of better quality throughout all iterations.

ACKNOWLEDGMENT

The authors would like to thank Robert van Amerongen for his many contributions, providing hands-on experience, and invaluable insight on the subject of load flow analysis, and UCTE/ENTSO-E for providing the UCTE study model, and their assistance on using it.

REFERENCES

[1] P. Schavemaker and L. van der Sluis, *Electrical Power System Essentials*. Chichester: John Wiley & Sons, 2008.

[2] B. Stott, "Review of load-flow calculation methods," *Proceedings of the IEEE*, vol. 62, no. 7, pp. 916–929, 1974.

[3] W. F. Tinney and C. E. Hart, "Power flow solution by Newton's method," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-86, no. 11, pp. 1449–1449, 1967.

[4] W. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proceedings of the IEEE*, vol. 55, no. 11, pp. 1801–1809, 1967.

[5] B. Stott and O. Alsac, "Fast decoupled load flow," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-93, no. 3, pp. 859–869, 1974.

[6] R. A. M. van Amerongen, "A general-purpose version of the fast decoupled loadflow," *IEEE Transactions on Power Systems*, vol. 4, no. 2, pp. 760–770, 1989.

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

[8] R. Idema, D. J. Lahaye, and C. Vuik, "Load flow literature survey," Delft Institute of Applied Mathematics, Delft University of Technology, Report 09-04, 2009.

[9] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. New Jersey: Prentice Hall, 1983.

[10] A. J. Flueck and H. D. Chiang, "Solving the nonlinear power flow equations with an inexact Newton method using GMRES," *IEEE Transactions on Power Systems*, vol. 13, no. 2, pp. 267–273, 1998.

[11] F. de Len and A. Semlyen, "Iterative solvers in the Newton power flow problem: preconditioners, inexact solutions and partial Jacobian updates," *IEE Proc. Gener. Transm. Distrib*, vol. 149, no. 4, pp. 479–484, 2002.

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

[13] Y. Saad, *Iterative methods for sparse linear systems*, 2nd ed. SIAM, 2000.

[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] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. The Johns Hopkins University Press, 1996.

[16] 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.
[17] P. Sonneveld and M. B. van Gijzen, “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] H. A. van der Vorst and C. Vuik, “The superlinear convergence behaviour of GMRES,” *J. Comp. Appl. Math.*, vol. 48, pp. 327–341, 1993.
[20] R. S. Dembo and T. Steihaug, “Truncated-Newton algorithms for large-scale unconstrained optimization,” *Mathematical Programming*, vol. 26, pp. 190–212, 1983.
[21] 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.
[22] A. Hohmann, “Inexact Gauss Newton methods for parameter dependent nonlinear problems,” Ph.D. dissertation, Freie Universität Berlin, 1994.

Reijer Idema obtained his MSc in Applied Mathematics (Computational Science and Engineering specialization) at the Delft University of Technology in 2007. Currently he is a PhD student at the Numerical Analysis research group of the Delft Institute of Applied Mathematics, Delft University of Technology.

Domenico Lahaye obtained his MSc in Applied Mathematics at the Free University of Brussels in 1994, his post-graduate degree in Mathematics for the Industry at the Eindhoven University of Technology in 1996, and his PhD in Computer Science at the Catholic University of Leuven in 2001. After having held postions 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, as assistant professor in 2007.

Kees Vuik obtained his MSc in Applied Mathematics at the Delft University of Technology in 1982. After a short stay at the Philips Research Laboratories, he obtained his PhD in Mathematics at Utrecht University in 1988. Thereafter he became employed at the Delft University of Technology, where he holds the position of 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 obtained his MSc in Electrical Engineering at the Delft University of Technology 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 is employed as full professor of the Power Systems Department of the Delft University of Technology.