Review of Load-Flow Calculation Methods

BRIAN STOTT, MEMBER, IEEE

Invited Paper

***Abstract*—A survey is presented on the currently available numerical techniques for power-system load-flow calculation using the digital computer. The review deals with methods that have received widespread practical application, recent attractive developments, and other methods that have interesting or useful characteristics. The analytical bases, computational requirements, and comparative numerical performances of the methods are discussed. Attention is given to the problems and techniques of adjustments in load-flow solutions, and the suitabilities of various methods for modern applications such as security monitoring and optimal load flow are examined.**

I. Introduction

LOAD FLOW (or power flow) is the solution for the static operating condition of an electric-power transmission system, and is the most frequently performed of routine digital-computer power network calculations. Over the last 20 years an enormous amount of effort has been expended in research and development on the numerical calculation process. For instance, [1] gives a large but by no means exhaustive bibliography on the subject, comprising more than 200 "respectable" papers in the English language alone. Out of these, 134 have appeared as publications of the IEEE Power Engineering Society and its predecessor.

This review will clearly be unable to cover every aspect of the problem and every proposed solution algorithm, nor is it intended to compete as a catalog of references with other recent sources [1], [2]. The aim is to present the underlying principles and techniques of the popularly accepted approaches, those more recent methods that seem to offer particular promise, and a selection of other methods that contain ideas of practical or theoretical interest.

Perhaps the most recurrent question arising in the load-flow field is—which is the best method to choose for a given application? The answer is rarely easy. The relative properties and performances of different load-flow methods can be influenced substantially by the types and sizes of problems to be solved, by the computing facilities available, and by the precise details of implementation. Any final choice is almost invariably a compromise between the various criteria of goodness by which load-flow methods are to be compared with each other. Every such criterion is directly or indirectly associated with financial cost, in the actual calculation itself, in the engineering application, or in the computer hardware and software requirements.

Load-flow calculations are performed in power-system planning, operational planning, and operation/control. They are increasingly being used to solve very large systems, to solve multiple cases for purposes such as outage security assessment, and within more complicated calculations such as optimization and stability. Table I gives a brief summary of some of the main types of load-flow solutions currently in application, and the requirements imposed on the numerical processes. Each application requires a combination of the types shown, e.g., some forms of security assessment call for approximate, unadjusted, on-line, multiple-case solutions.

TABLE I

LOAD-FLOW CALCULATIONS—TYPES AND REQUIREMENTS

Types of Solution		
	Accurate	Approximate
	Unadjusted	Adjusted
	Off-Line	On-Line
	Single-Case	Multiple Cases
Properties Required of Load-Flow Solution Method		
High speed	especially for:	large systems real-time applications multiple cases interactive applications
Low storage	especially for:	large systems computers with small core-store availability
Reliability	especially for:	ill-conditioned problems outage studies real-time applications
Versatility	ability to handle conventional and special features (adjustments, representation of power-system apparatus); suitability for incorporation into more complicated processes	
Simplicity	ease (and cost) of coding, maintaining, and enhancing the algorithm and computer program based on it.	

A. Brief History of Load Flow

Prior to, and for some time after, the advent of digital computers, load-flow solutions were obtained using network analyzers. The first really practical automatic digital solution methods appeared in the literature in 1956 and subsequently [10]–[12]. These Y-matrix iterative methods were well-suite to the early generations of computers since they require minimal computer storage. Although they perform satisfactorily on many problems, they converge slowly, and too often not at all. The incentive to overcome this deficiency led to the development of Z-matrix methods [19]–[21], which converge more reliably but sacrifice some of the advantages of Y-matrix iterative methods, notably storage and speed when applied to large systems. Around the same time, the Newton(–Raphson) method was shown to have very powerful convergence properties [24], [25], but was computationally uncompetitive. Major breakthroughs in power-system network computation came in the mid-1960's, with the development by Tinney and others of very efficient sparsity-programmed ordered elimination [3]. One of its earliest successes was in dramatically improving the computing speed and storage requirements of Newton's method, which has now come to be widely regarded as the preeminent general-purpose load-flow approach [26], and has been adopted by much of

Manuscript received December 14, 1973; revised January 21, 1974.
The author is with the University of Waterloo, Waterloo, Ont., Canada, on leave from the University of Manchester Institute of Science and Technology, Manchester, U. K.

industry. Currently, with the stimulus of increasing problem sizes, on-line applications, and system optimization, newer methods are emerging which are also expected to find wide application.

II. Notations

A. Complex Quantities at Bus i

$E_i = V_i/\underline{\theta_i} = e_i + jf_i$	Nodal voltage.
I_i	Nodal injected current.
$S_i = P_i + jQ_i$	Net nodal injected power.
$\Delta S_i = \Delta P_i + j\Delta Q_i$	Power mismatch.
ΔI_i	Current mismatch.
$\Delta E_i = \Delta V_i/\underline{\Delta\theta_i}$	Correction to nodal voltage.

B. Matrices

$Y = G + jB$	Nodal admittance matrix.
Z	Nodal impedance matrix.
J	Jacobian matrix.

C. General

n	Number of buses in system.
X_{ik}	Reactance of branch between buses i and k.
$\theta_{ik} = \theta_i - \theta_k$	
s	Slack bus index.

Superscripts ^{sp}, ^{cal}, and * denote specified value, calculated value, and complex conjugate, respectively.

Subscripts $_G$ and $_L$ denote generation and load, respectively.

All unsubscripted upper-case quantities denote vectors or matrices.

$k\omega i$ denotes a bus k directly connected to bus i.

$k\mu i$ denotes a PQ-type bus directly connected to bus i.

$k \in i$ is the same as $k\omega i$, except that it includes the case $k = i$.

III. Basic Analytical Formulation of Load-Flow Problem

The basic formulation of the load-flow problem is now well-known [4]–[7], and is presented only briefly in this section. Virtually all load-flow calculation methods have been devised initially for the solution of this basic problem. Extensions to it are usually "grafted on" afterwards, and Section IX discusses some of them.

A. Basic Equations and Bus Types

A balanced three-phase power system is assumed, and the transmission system is represented by its positive-phase-sequence network of linear lumped series and shunt branches. Nodal analysis is almost universally preferred in this application, and leads to the network nodal admittance matrix equation

$$I = Y \cdot E \tag{1}$$

where matrix Y is square, sparse, and symmetrical (in the absence of phase shifters or mutual couplings represented by nonbilateral network branches).

The static operating state of the system is specified by the constraints on power and/or voltage at the network buses. Buses are categorized into three types, as follows:

A *PQ bus* i is one at which the total injected power is specified:

$$S_i^{sp} = P_i^{sp} + jQ_i^{sp} = P_{Gi}^{sp} - P_{Li}^{sp} + j(Q_{Gi}^{sp} - Q_{Li}^{sp}) = E_i I_i^*. \tag{2}$$

A *PV bus* i is one at which the total injected active power is specified, and the voltage magnitude is maintained at a specified value by reactive-power injection. The bus constraints are

$$P_i^{sp} = P_{Gi}^{sp} - P_{Li}^{sp} = \mathrm{Re}\,(E_i I_i^*) \tag{3}$$

$$V_i^{sp} = (e_i^2 + f_i^2)^{1/2} = V_i = |E_i|. \tag{4}$$

The system *slack (or swing) bus* is a fictitious concept, created by the load-flow analyst. It arises because the system I^2R losses are not known precisely in advance of the load-flow calculation. Therefore, the total injected active power cannot be specified at every single bus. It is customary to choose one of the available voltage-controlled buses as slack, and to regard its active power as unknown. The difference between its expected and solved megawatt output then represents the error in the prior estimate of system I^2R losses. If the slack bus is a point of large generation, this error will be a small proportion of the net bus generation. The slack-bus voltage angle is usually assigned as the system complex phase reference, and its complex voltage E_s is therefore known.

B. Bus Mismatches and Solution Accuracy Criteria

An exact load-flow solution is achieved when the bus constraints (2)–(4) are satisfied. Nearly all load-flow algorithms are devised to satisfy (4) exactly, and in these cases, the degree to which the specified and calculated bus powers in (2) and (3) equate with each other constitutes an accuracy test for the solution.

For each PQ bus i the complex power mismatch is derived from (2) and (1) as

$$\Delta S_i = S_i^{sp} - E_i I_i^* = P_i^{sp} + jQ_i^{sp} - E_i \sum_{k \in i} Y_{ik}^* E_k^*. \tag{5}$$

Equation (5) can be separated into its real and imaginary parts, giving active- and reactive-power mismatch expressions relevant to both (2) and (3). Rectangular- and polar-coordinate versions are presented as follows.

For each PQ or PV bus i:

$$\Delta P_i = P_i^{sp} - \mathrm{Re}\left\{(e_i + jf_i) \sum_{k \in i} (G_{ik} - jB_{ik})(e_k - jf_k)\right\} \tag{6a}$$

$$= P_i^{sp} - V_i \sum_{k \in i} (G_{ik} \cos\theta_{ik} + B_{ik} \sin\theta_{ik}) V_k. \tag{6b}$$

For each PQ bus i:

$$\Delta Q_i = Q_i^{sp} - \mathrm{Im}\left\{(e_i + jf_i) \sum_{k \in i} (G_{ik} - jB_{ik})(e_k - jf_k)\right\} \tag{7a}$$

$$= Q_i^{sp} - V_i \sum_{k \in i} (G_{ik} \sin\theta_{ik} - B_{ik} \cos\theta_{ik}) V_k. \tag{7b}$$

Current-mismatch expressions are also available. From (5) we can define

$$\Delta I_i = \Delta S_i^*/E_i^* = (P_i^{sp} - jQ_i^{sp})/E_i^* - \sum_{k \in i} Y_{ik} E_k. \tag{8}$$

The most common convergence criterion used in practice is

$$\Delta P_i \le c_p, \qquad \text{for all } PQ \text{ and } PV \text{ buses } i$$

$$\Delta Q_i \le c_q, \qquad \text{for all } PQ \text{ buses } i$$

where c_p and c_q are tolerances chosen typically in the range

0.01 to 10 MW/mvar. These tolerances put an approximate upper bound on the errors in any calculated line flows, and therefore have a direct engineering interpretation. Alternative more complicated criteria, e.g., using the sum of the absolute values or squares of the mismatches, are probably superfluous. Bus voltage-change tests are often used for load-flow algorithms in which mismatches are not readily available. Such tests are sensitive to the convergence rate of the solution process, and are usually used as initial stopping criteria, after which the mismatches are computed and tested.

IV. Y-Matrix Iterative Methods

A. Relaxation and the Gauss–Seidel Method

The Y-matrix iterative methods of load-flow calculation are based on the iterative solution of the linear equation (1) for the bus voltages, using the relaxation algorithm [8]. Assuming temporarily that the bus currents I are known, the current mismatch (residual) at each bus is, from (1):

$$\Delta I_i = I_i - \sum_{k \in i} Y_{ik} E_k. \tag{9}$$

Each application of the relaxation algorithm liquidates ΔI_i by reevaluating E_i. The usual computational form of the algorithm is

$$E_i = \left(I_i - \sum_{k \neq i} Y_{ik} E_k \right) \Big/ Y_{ii}. \tag{10a}$$

An alternative form, expressed in terms of the change in E_i, is

$$\Delta E_i = \Delta I_i / Y_{ii}. \tag{10b}$$

By far the most popular way of applying algorithm (10) is to use a systematic, single-sweep, successive-displacements mode of iteration, i.e., the Gauss–Seidel method, since this has the attractions of simplicity, comparatively good performance, and no need to store previous values. The name Gauss–Seidel refers specifically to the above numerical method, but is sometimes used erroneously in the load-flow literature to be synonymous with "successive displacements."

The matrix theory shows that the method converges if the largest eigenvalue-modulus ρ of its iteration matrix (a matrix derived from Y) is less than unity [9]. A more useful sufficient, though over-stringent, condition for convergence is that Y should possess strict diagonal dominance. This condition may not be satisfied for practical power networks, but the self-admittances of the buses are usually large relative to the mutual admittances, and convergence is obtained. The "spectral radius" ρ is associated with the least diagonally dominant row (or column) of Y, which gives the means for identifying network characteristics that affect convergence. Junctions of very high and low series impedances, and large capacitances, encountered with cable circuits, long EHV lines, series and shunt compensation, are detrimental to convergence because they weaken diagonal dominance.

B. The Slack Bus

For a load-flow calculation, the bus constraints have to be incorporated into the relaxation algorithm (10). As far as the slack bus is concerned, its complex voltage is known, and therefore it is simply not reevaluated. This is equivalent to reducing the order of (1) by deleting the slack bus row and column from Y, and thus ρ is changed. The choice of slack bus can affect convergence considerably in difficult cases—by removing the least diagonally dominant row and column of Y it is possible to reduce ρ.

C. Glimn and Stagg Method

In this, the most straightforward and standard load-flow method [11], the nonlinear bus constraints are incorporated by substituting for the unknown I_i in (10) from (2). For each PQ bus, the algorithms (10a) and (10b), respectively, become

$$E_i = \left\{ (P_i^{sp} - jQ_i^{sp})/E_i^* - \sum_{k \neq i} Y_{ik} E_k \right\} \Big/ Y_{ii} \tag{11a}$$

and

$$\Delta E_i = \Delta I_i / Y_{ii} = \Delta S_i^* / E_i^* Y_{ii} = (\Delta P_i - j \Delta Q_i)/E_i^* Y_{ii}. \tag{11b}$$

The problem is now "weakly linear," with the degree of nonlinearity determined by the variation during the solution of the denominator term E_i^*. The factors affecting convergence in the purely linear problem (10) remain generally valid unless this variation is large.

For a PV bus, V_i is specified and Q_i^{sp} is not available. In (11a), Q_i^{sp} can be replaced by Q_i^{cal}, the calculated reactive power, which is the same as setting ΔQ_i in (11b) to zero. After reevaluating E_i, its magnitude V_i is reset to V_i^{sp} while maintaining its new angle. In this strategy, the angular change $\Delta\theta_i$ in E_i is being strongly associated with ΔP_i (since $\Delta Q_i = 0$), as seen from (11b). A minor variant employs the corresponding Q–V association principle. Starting with the value Q_i^{cal}, Q_i in (11a) is adjusted by linear extrapolation until E_i takes on the specified magnitude [13].

D. Zero-Mismatch Methods

In the nonlinear load-flow problem, each application of (11) does not entirely liquidate ΔI_i or ΔS_i. Several variants make more effort to satisfy the bus power constraints at each iteration. In (11a) an inner iteration loop for E_i can be created by substituting the newly reevaluated E_i from the left-hand side back into the denominator term E_i^* on the right-hand side. If iterated to convergence, this inner-loop process succeeds in liquidating ΔS_i. An alternative way of doing the same thing is to construct and solve a pair of real quadratic equations in e_i and f_i from the left-hand E_i and right-hand E_i^* terms [14]. For a PV bus one of these equations is replaced by the magnitude constraint

$$(V_i^{sp})^2 = e_i^2 + f_i^2. \tag{12}$$

For certain difficult systems, this gives better convergence, but otherwise the extra computation is not justified.

E. Ward and Hale Method

This method [10] solves linearized versions of the previously mentioned quadratic equations at each iteration. The solution, for Δe_i and Δf_i, constitutes one iteration of a local Newton solution. The method performs very similarly to the Glimn and Stagg version, from which it differs only slightly. It was derived in the original paper from the incremental expression, for PQ buses,

$$P_i^{sp} + jQ_i^{sp} = E_i(I_i + Y_{ii}\Delta E_i)^* + \Delta E_i I_i^* \tag{13}$$

which differs from (11) only in the last term $\Delta E_i I_i^*$.

F. Polar Formulation

Y-matrix iterative methods are coded most efficiently with the complex voltages in rectangular form. A polar version, which can be shown to approximate to (11b) under the assumptions that $G_{ii} \ll B_{ii}$ and that $\Delta\theta_i$ and ΔV_i are small, is

$$\Delta\theta_i = -\Delta P_i / B_{ii} V_i^2 \tag{14a}$$

$$\Delta V_i = -\Delta Q_i / B_{ii} V_i. \tag{14b}$$

This approach, either in the Hubert and Hayes implementation [15], or in a more conventional successive-displacements iteration mode, does not appear to offer any advantages over the rectangular forms [1].

G. Secondary-Adjustment Method [16], [17]

Whenever algorithm (11) is applied to bus i, the resulting voltage change ΔE_i will in part be responsible for the voltage change at each connected bus k within the next iteration cycle. If bus k is a PQ type, an approximate relation between ΔE_i and the change ΔE_k due to ΔE_i is

$$\Delta E_k = -\Delta E_i Y_{ki} / Y_{kk}. \tag{15}$$

This correction transmittal can be anticipated in advance by substituting for E_k on the right-hand side of (11a) the voltage $E_k + \Delta E_k$, where ΔE_k is given by (15).

The resulting algorithm, in the forms (11a) or (11b), turns out to be unchanged, except that the normal admittance term Y_{ii} used in it is replaced by

$$Y_{ii}^{\text{eff}} = Y_{ii} - \sum_{k \mu i} Y_{ik} Y_{ki} / Y_{kk}. \tag{16}$$

Note that PV buses connected to bus i are omitted from (16). The final iteration scheme is to apply (11) but with Y_{ii}^{eff} and then to adjust the voltage of each connected PQ bus using (15). The method gives faster and more reliable convergence for most systems [1], except those containing many PV buses. For well-conditioned systems, the reduction in the number of iterations compared with the normal algorithm (11) is offset by the extra computation. Its added reliability can be a great asset, however.

H. Acceleration

Y-matrix iterative methods converge slowly, because of the loose mathematical coupling between the buses—at each iteration cycle, an improvement in each bus voltage can only affect the voltage improvements of the buses directly connected to it. Acceleration techniques are invariably used in practice to speed up the convergence.

The most popular acceleration method is successive overrelaxation (SOR), which can be explained using (11b). A fixed empirically determined acceleration factor α $(1 < \alpha < 2)$ is applied to each voltage-change reevaluation thus:

$$\Delta E_i = \alpha \Delta S_i^* / E_i^* Y_{ii} \tag{17}$$

and can be interpreted as a small linear extrapolation of E_i. With a good choice of α the convergence rate is usually improved by a factor of two or more, compared with no acceleration ($\alpha = 1$), and an otherwise divergent case can sometimes be made to converge. For a given system, it is often found that a near-optimal choice of α remains valid over a range of operating conditions. Although a complex value of α can be used, it is more normal to use a real value, or to accelerate the real and imaginary parts of ΔE_i with slightly different values of α [11].

Several attempts have been made to devise variable rather than fixed small-extrapolation methods [15], [18]. Introducing the iteration index p, one such method [18] uses instead of (17) the additive factor β rather than the multiplicative factor α:

$$\Delta E_i^{(p)} = (\Delta S_i^{(p)} + \beta_i^{(p)})^* / E_i^{*(p)} Y_{ii} \tag{18}$$

where

$$\beta_i^{(p)} = b \Delta S_i^{(p-1)} + b^2 \Delta S_i^{(p-2)} + b^3 \Delta S_i^{(p-3)} + \cdots$$

and b is a fixed "boost" factor usually in the range 0.75 to 0.92. Hence a bus whose previous convergence is poor (large mismatches) will receive a correspondingly large extrapolation. Compared with SOR, this method gives faster convergence on some systems but not on others, and rarely justifies the extra computation at each iteration [1].

Large nonlinear extrapolation can be used if at some stage the iterative process is detected to be converging sufficiently monotonically [12], [13]. Aitken's Δ^2 process is based on the assumption of linear or geometric convergence, i.e., that

$$\frac{E_i^{\text{sol}} - E_i^{(p)}}{E_i^{\text{sol}} - E_i^{(p-1)}} = \frac{E_i^{\text{sol}} - E_i^{(p-1)}}{E_i^{\text{sol}} - E_i^{(p-2)}}. \tag{19}$$

Provided that the necessary previous values have been stored and the test for monotonic convergence of all voltages is satisfied, (19) can be solved for the "exact" solution E_i^{sol}. In practice, of course, convergence is not strictly linear, and this is not the true solution, and the iterations must be continued. This form of acceleration can be used concurrently with SOR, and it is usually applied automatically several times during the calculation. The computational overheads are high, and most often not justified by the reduction in the number of iterations.

I. Computational Efficiency of Y-Matrix Iterative Methods

Computationally, the salient feature of these methods is that the number of elements in the summation term in (10a) is small, averaging about three for well-developed power-system networks. The off-diagonal elements of the Y-matrix, for use in the summation, are therefore stored and addressed compactly, often taking advantage of Y-matrix symmetry. Both the storage requirements for the network and the computation per iteration are then small, and roughly proportional to the number of buses n. Since the number of iterations for a large well-conditioned system is of order n, the total iterative computing time varies approximately with n^2. As the size of problems to be solved increases, Y-matrix iterative methods become less and less competitive with newer methods such as Newton's. However, their storage requirements are very low. The Y-matrix compacting takes little computation, and efficient coding of the methods is relatively very easy.

V. Z-Matrix Methods

A. Z-Matrix Algorithms [19]–[22]

The major difference in principle between the Y-matrix iterative methods and the Z-matrix methods is that in the latter, the linear equation (1) is solved directly for E in terms of I using the inverse of Y:

$$E = Y^{-1} \cdot I = Z \cdot I. \tag{20a}$$

An alternative version eliminates the equation for the slack bus from (1) before inverting Y. This leads to the $(n-1)$th order equation

$$E = Z \cdot I + C \cdot E_s \tag{20b}$$

where matrix Z is different from the one in (20a), E and I do not now contain E_s and I_s, respectively, and C is a constant vector.

The basic scheme in either version incorporates the bus constraints by substituting for currents, just as in Section IV-C. Each bus current I_i $(i \neq s)$ is evaluated from

$$I_i = (P_i^{sp} - jQ_i)/E_i^* \tag{21}$$

where $Q_i = Q_i^{sp}$ for a PQ bus, and $Q_i = Q_i^{cal}$ for a PV bus (equivalent to setting $\Delta Q_i = 0$). The bus voltages are reevaluated by the iterative application of (20) and (21), taking the angular voltage change only for a PV bus. Using version (20a), it is best to update I_s at every stage to reflect the changes in the bus voltages [22]. Versions (20a) and (20b) are then numerically identical except for rounding errors.

The conventional Z-matrix approach uses the more powerful successive-displacements iteration mode, and therefore, by definition, the nonsparse explicit impedance matrix Z as shown in (20). A technique of great general value is to represent a portion of each bus load as a fixed shunt admittance, which in effect reduces the value of the nonlinear bus power constraint [22]. A number of variants on the Z-matrix algorithm are available [22]. As in Section IV-D, zero-mismatch techniques can be used, either by reiterating each voltage in an inner loop between (20) and (21) or by solving a pair of real quadratic equations [20]. This usually improves the speed and reliability of convergence.

B. The Impedance Matrices

The driving-point and transfer impedance matrices Z in (20a) and (20b) are referred to ground and the slack bus, respectively. In the former case, it is usually necessary to create at least one strong artificial tie to ground to avoid numerical difficulties when obtaining Z, because in the absence of this, Y is ill-conditioned or even singular. A large shunt admittance inserted at the slack bus most simply achieves the desired effect.

The impedance matrix is conventionally computed by an ordered bus-by-bus assembly process, reflecting the shunt and series branches into the formative matrix as soon as possible using the inverse-matrix modification algorithm [21]. More modern sparsity-oriented inversion algorithms are probably faster [9].

C. Convergence Properties

By virtue of the direct network solution, each bus voltage is coupled with all bus currents. Convergence is therefore rapid and reliable, compared with Y-matrix iterative methods, and typically takes 8 to 20 iterations for medium or large systems, but sometimes much more.

Systems with no PV buses tend to converge best. To explain this phenomenon, consider in (20) the reevaluation of any bus voltage E_i, by multiplying the relevant row of Z with the vector I. This process is unable to distinguish whether each of the currents used belongs to a PV or a PQ bus. In other words, it fails to transmit to E_i the information that the magnitude of each PV-bus voltage E_k $(k \neq i)$ is to be held fixed, i.e., that the equation in (20) for E_k is in reality an equation in angle only. The simultaneous-displacements version of the Z-matrix method is particularly badly affected by the presence of PV buses, although it works well without them. This is unfortunate, because the sparse triangular factors of matrix Y can be used instead of Z in this version, with a great saving in storage and computation per iteration (see Section VIII-C).

As with all load-flow methods, a rigorous convergence analysis is not easy. The incremental form of the algorithm is

$$\Delta E = Z \cdot \Delta I. \tag{22}$$

In the simultaneous-displacements version, an approximate expression for ΔE at iteration p is

$$\Delta E^{(p)} = K \cdot \{Z \cdot [P^{sp} - jQ]\}^p \cdot E^{(0)} \tag{23}$$

where $[P^{sp} - jQ]$ is a diagonal matrix. For convergence, $\Delta E^{(p)}$ must tend to zero as p tends to infinity. Therefore, convergence is best if the bus powers are small (hence the success of representing loads as shunt admittances), and if the spectral radius ρ of Z is small. The Z-matrix method is not usually too sensitive to the choice of slack bus. Nevertheless, a good choice is the bus whose row sum in Z is largest, since this in effect reduces ρ [23].

D. Computational Efficiency of Z-Matrix Methods

The dominating feature of Z-matrix methods is the need to obtain, store, and iterate with the nonsparse Z matrix. Even utilizing symmetry, more than n^2 real words of storage are required for it, which can make entirely in-core solutions unattractive or impossible, depending on the sizes of the problem and the computer. Block transfers of rows of Z to and from backing store may have to be programmed.

The initial construction of Z by the assembly method takes an amount of computing of order between n^2 and n^3, depending on the network configuration. The Z matrix of a given system can be stored for use from study to study. Each individual network change by inverse-matrix modification takes $2n^2$ real operations (utilizing symmetry). More than $4n^2$ real operations are performed at each iteration cycle.

For problems that can be solved by both, Z-matrix methods are rarely competitive with Y-matrix iterative methods, and therefore certainly not with modern methods. Computation and storage become astronomical for very large problems.

VI. Newton Methods

A. The Newton Algorithm

The generalized Newton(–Raphson) method is an iterative algorithm for solving a set of simultaneous nonlinear equations in an equal number of unknowns $F(X) = 0$. At a given iteration point, each function $f_i(X)$ is approximated by its tangent hyperplane. This linearized problem is constructed as the Jacobian-matrix equation

$$F(X) = -J \cdot \Delta X \tag{24}$$

which is then solved for the correction ΔX. The square Jacobian matrix J is defined by $J_{ik} = \partial f_i / \partial x_k$, and represents the slopes of the tangent hyperplanes. Matrix J is highly sparse in the load-flow application, and (24) is solved directly and rapidly by sparsity-programmed ordered triangulation and back-substitution [3].

The Newton method's quadratic convergence is faster than that of any other load-flow approach, and the process "homes in" very rapidly when the solution point is close. Its

performance is sensitive to the behaviors of the functions $F(X)$ and hence to their formulation. The more linear they are, the more rapidly and reliably Newton's method converges. Nonsmoothness, i.e., humps, in any function $f_i(X)$ in the region of interest can cause convergence delays, total failure, or misdirection to a nonuseful solution.

Since the chosen load-flow functions $F(X)$ tend not to be too nonlinear, and reasonably good initial estimates are available, these difficulties are encountered infrequently. In fact, applied to the vast majority of practical load-flow problems, Newton's method is very reliable and extremely fast in convergence. Its sensitivity is minimal to factors causing poor convergence with many other load-flow methods, such as choice of slack bus and series capacitors.

The Newton load-flow formulations adopted to date use for $F(X)$ the bus power or current mismatch expressions, and designate the unknown bus voltages as the problem variables X. Mathematically speaking, the complex load-flow equations are nonanalytic, and cannot be differentiated in complex form. In order to apply Newton's method, the problem is separated into real equations and variables. Rectangular or polar coordinates may be used for the bus voltages.

B. *The Polar Power-Mismatch Version* [24]–[26]

This is the most widely used of all formulations, whose Jacobian-matrix equation (24) can be written for convenience of presentation in the partitioned form:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} H & N \\ M & L \end{bmatrix} \cdot \begin{bmatrix} \Delta\theta \\ \Delta V \end{bmatrix} \tag{25}$$

Slack-bus mismatches and voltage corrections are not included in (25), and likewise ΔQ_i and ΔV_i for each PV bus are absent. The order of the equation is therefore $(n_{pv}+2n_{pq})$. The submatrices H, N, M, and L represent the negated partial derivatives of (6b) and (7b) with respect to the relevant θ's and V's, e.g., $H_{ik}=-\partial\Delta P_i/\partial\theta_k$. If buses i and k are not directly connected, their "mutual" terms in the J matrix are zero, and J is thus highly sparse, with positional but not numerical symmetry. In (25), each right-hand correction ΔV_i is usually divided by V_i, to gain computationally by making the terms in M and L similar to those in N and H, respectively. This does not alter the numerical performance of the method.

The polar power-mismatch version converges to high accuracy nearly always in 2 to 5 iterations from a flat start ($V=1$ per-unit, $\theta=0$), independent of system size. The accepted formulation (25) can be improved by a minor modification, which very often reduces the number of iterations by one, and can avoid divergence in some extreme cases. Noting that the performance of Newton's method is closely associated with the degree of problem nonlinearity, the best left-hand defining functions are the most linear ones. If (7b) is divided throughout by V_i, only one term Q_i^{sp}/V_i on the right-hand side of this equation is not linear in V. Moreover, for practical values of Q_i^{sp} and V_i, this nonlinear term is numerically relatively small. It is therefore preferable to use a problem-defining function $\Delta Q/V$ on the left-hand side of (25) in place of ΔQ. Dividing ΔP by V can also be helpful, but has a relatively small effect, since the active-power component of the problem is not strongly coupled with voltage magnitudes.

C. *Rectangular Power-Mismatch Version* [25]

This version uses the real and imaginary parts of the bus voltages as variables, and hence derives from (6a) and (7a). The number of equations and variables is greater than that for (25), by the number of PV buses. A voltage-magnitude-squared mismatch equation from (12) is needed for each PV bus. With sparsity programming, this increase in order is of little significance. Indeed, each iteration is marginally faster than for (25) because there are no time-consuming trigonometrical functions (noting though that even the polar version avoids these as far as possible by using rectangular arithmetic in constructing (25)) [26]. From a series of tests carried out by the author, the rectangular version seems to be slightly less reliable and rapid in convergence than the polar version.

By neglecting all the "mutual" terms in the Jacobian matrix, a simultaneous-displacements version of the Ward and Hale method is obtained.

D. *Current-Mismatch Versions*

Rectangular or polar coordinate Newton versions can be constructed from the real and imaginary parts of the complex current-mismatch equation (8). In both versions, it is convenient to revert to the power-mismatch equation (6) for a PV bus, adding a voltage-squared mismatch equation if (6a) is chosen. These versions generally perform less satisfactorily than the power-mismatch methods. The main interest lies with the rectangular version. For PQ buses, its J-matrix "mutual" terms are simply network branch admittances, and on no-load the "self" terms become likewise. The version can thus perform relatively reliably for difficult light-load problems [27], [28]. Note that for systems with no PV buses and on no-load, the Jacobian-matrix equation is exactly equivalent to the Z-matrix equation (22).

E. *Algorithmic Enhancements*

A number of schemes are available for attempting to improve the performance of Newton's method [27]. One of the simplest of these is to impose limits on the permissible sizes of the voltage corrections at each iteration, thereby helping to negotiate humps in the defining functions. If the limits are permanently applied they should not be too small, otherwise convergence will be slowed down for well-behaved problems. A better approach is to back-track as soon as divergence is seen to have started, and then apply small limits.

With its quadratic convergence, Newton's method takes maximal advantage of good initial voltage estimates. As with all other load-flow methods, previously stored solutions can sometimes be used as starting values. Some programs perform one or two Gauss–Seidel iterations before the Newton process [26]. This is beneficial provided that the relatively weak Gauss–Seidel method does not diverge when faced with a difficult problem. A most rapid and reliable Newton program can be created by calculating good initial angular estimates using the dc load flow (see Section X-A) and also good voltage-magnitude estimates by a similar technique [29].

Iteration time can be saved by using the same triangulated Jacobian matrix for two or more iterations [26]. However, the lower triangular factor of J now needs to be stored, and the algorithm loses some speed and reliability of convergence.

F. *Construction and Solution of Jacobian-Matrix Equation*

For nonsmall systems (say, larger than 20 buses) Newton's method relies for its computational efficiency on skillful programming in the construction and solution of the Jacobian-matrix equation. The remarks on procedure in this section are broadly applicable to all Newton versions, but specific details refer to the standard version (25).

Prior to iteration, the system buses are reordered to minimize fill-in during the triangulation process later. The two mismatches and the two voltage corrections for each PQ bus are ordered consecutively to make computational use of common terms in constructing (25), to ensure large matrix diagonal elements, and to preserve positional symmetry of J [26]. Row (Crout) elimination for the upper triangular reduction is most efficient in sparsity programming. A choice of approaches is then available: a) the rows (or row pairs) of (25) are computed and eliminated one at a time, or b) the complete mismatch vector and compact matrix J are computed together, enabling further computational use to be made of common terms in the mismatch and J-matrix expressions, and enabling convergence testing to be performed on the mismatches before deciding whether another solution of (25) is necessary.

Approach b) requires one less solution of (25) than approach a), but needs more storage if the number of elements in the J matrix is larger than the number in its upper triangular reduction. Unless the power-system network is very large or complicated, and if a standard dynamic ordering method is used [3], this is usually the case.

G. Computational Efficiency of Newton Methods

Although programming technique is important in all load-flow methods for obtaining fast execution and storage economy, it is the cornerstone of methods such as Newton's that use ordered elimination for solving the large sparse matrix equations [3], [26]. These methods realize their full computational potential only if the programming is of the very highest standard. Packages for solving a given sparse network-matrix equation such as the Newton equation (25) are becoming generally available, but the need to construct the equation compactly and efficiently remains.

If these programming requirements are fully satisfied, then the computing time per iteration of Newton's method rises little more than linearly with the number of buses in the system, on average. This is an empirical function of the network topology and the success of the bus-ordering scheme. Since the number of iterations is size-invariant, the superiority of Newton's method speedwise over previous methods increases rapidly as the size of the system to be solved increases. For typical large systems, the computing time for one Newton iteration is roughly equivalent to seven Gauss–Seidel iterations [26]. The usually satisfactory simple dynamic bus-ordering process takes less time than one Newton iteration. If therefore we take the example of a 500-bus problem that takes 500 iterations using Gauss–Seidel, and four iterations (to a higher accuracy) using Newton, the speed advantage of the latter for the iterative calculation is about 15:1.

Like speed, Newton's storage requirements are not exactly predictable, because the fillup in the sparse upper triangular reduction of the J matrix is not known in advance. They also depend on the speed–storage tradeoff decisions made by the programmer. However, in the approaches most commonly adopted by industry, a practical upper limit for the storage required for the J-matrix equation and its solution is about $18n$ real words plus their addressing integers. Approach a) of Section VI-F will usually require less. With large modern computers, this extra storage compared with, say, the Y-matrix iterative methods, does not prevent very large systems from being solved in core.

VII. Decoupled Methods

A. The Decoupling Principle

An inherent characteristic of any practical electric-power transmission system operating in the steady state is the strong interdependence between active powers and bus voltage angles, and between reactive powers and voltage magnitudes. Correspondingly, the coupling between these "P–θ" and "Q–V" components of the problem is relatively weak. Applied numerical methods are generally at their most efficient when they are able to take advantage of the physical properties of the system being solved. In the load-flow problem, there has been a recent trend towards this objective by "decoupling" (solving separately) the P–θ and Q–V problems.

B. Decoupled Newton Methods

In any formal Newton method, half of the elements of the Jacobian matrix represent the weak coupling referred to above, and therefore have relatively small numerical values. These elements may be neglected. Any such approximations to J inevitably sacrifice the true quadratic convergence property, but compensating computational benefits can accrue [33].

Many algorithmic possibilities emerge from this. For brevity, only one recommended "good" decoupled Newton version is presented here. In (25) the elements to be neglected are those contained in submatrices N and M. Equation (25) is then separated into two smaller matrix equations, viz., the P–θ and Q–V problems. Adopting the minor formulation modifications of Section VI-B that can improve the polar power-mismatch method, the decoupled Newton equations become

$$\Delta P/V = A \cdot \Delta\theta \tag{26a}$$

and

$$\Delta Q/V = C \cdot \Delta V \tag{26b}$$

where A and C are negated Jacobian matrices. In this method, dividing ΔQ by V is important, since it substantially reduces the nonlinearity of the Q–V problem. An alternative but not superior version achieves a similar effect by replacing $\Delta Q/V$ by Im (ΔI) [34].

Equations (26a) and (26b) can be constructed and solved simultaneously with each other at each iteration. A much better approach is to conduct each iteration cycle by first solving (26a) for $\Delta\theta$, and use the updated θ in constructing and then solving (26b) for ΔV. Each of these construction/solutions can be performed alternately in the same storage area. The advantages of this block-successive iteration scheme are apparent from the beginning of the solution process. The first calculated values of θ are accurate to within a few degrees, even when starting from $\theta = 0$. The first solution of (26b) then usually gives remarkably good values for V, within say 0.3 percent of the final solution. Subsequent iterations refine these values, with the θ's taking longer to converge than the V's.

In its form (26) the decoupled method converges at least as reliably as the formal Newton version (25) from which it was derived. For convergence to practical accuracies, it usually takes a similar number of iterations. For very high accuracies it takes more iterations, because overall quadratic convergence is lost. However, it should be observed that the accuracy "overkill" produced by the final iteration of a formal

Newton solution is a bonus that has no practical value for most applications.

The decoupled Newton version saves by a factor of four on the storage for the J matrix and its triangulation. This is of course only a proportion of the total core required by a complete program, and the overall saving is more in the region of 35 to 40 percent. The computation per iteration is 10 to 20 percent less than for the formal Newton method.

C. Fast Decoupled Method

A variety of contributions have led to the evolution of this method [29], [33]–[37]. In its fully developed form [38], it is derived from (26) by making constant approximations to matrices A and C. Equation (26) is then transformed, in per-unit, to

$$\Delta P/V = B' \cdot \Delta\theta \tag{27a}$$

and

$$\Delta Q/V = B'' \cdot \Delta V \tag{27b}$$

where

$$B_{ik}' = -1/X_{ik} \quad (i \neq k),$$

$$B_{ii}' = \sum_{k \omega i} 1/X_{ik}, \quad \text{and} \quad B_{ik}'' = -B_{ik}.$$

Matrices B' and B'' represent constant approximations to the slopes of the tangent hyperplanes of the functions $\Delta P/V$ and $\Delta Q/V$, respectively. In effect, they are very close to being the Jacobian matrices A and C, evaluated at system no-load. The fixed-tangent iteration method [8] embodied in (27a) and (27b) is not vulnerable to humps under loaded system conditions in the left-hand problem-defining functions.

Pursuing the decoupling principle to its logical conclusion, network elements that primarily affect the Q–V problem (e.g., shunt susceptances and transformer off-nominal taps) should not be represented in B'. Similarly, phase shifts should not be represented in B''. In fact, the method has been found to work just as well if phase shifts are left out of B' also. Consequently, both B' and B'' are always symmetrical, and their constant sparse upper triangular factors are calculated and stored once only at the beginning of the solution. To solve (27a) and (27b), forward and backward substitutions are performed using these factors.

The method converges very reliably, usually in 2 to 5 iterations for practical accuracy on large systems. The method has the decoupled property of giving a very good approximate solution after the first one or two iterations. Provided that the $\Delta P/V$ and $\Delta Q/V$ functions are calculated efficiently, the speed per iteration is roughly 5 times that of the formal Newton method, and two-thirds that of the Gauss–Seidel methods. The storage requirements for the B' and B'' factors are in between the corresponding J-matrix requirements of the formal and decoupled Newton methods. Using a standard triangulation package, programming is easier than the Newton methods, whose compact J matrices have to be constructed at each iteration.

A further algorithmic development may sometimes find useful application. For systems with only one infeed point (no PV buses), matrices B' and B'' have exactly the same structure. Provided that the system does not contain significantly large shunt susceptances (which for best convergence should be excluded from B'), B' and B'' have almost the same values. Therefore only one B matrix need be used, thus saving storage and triangulation time. Even PV buses can be catered for using this single-B approach [39]. For each such bus, ΔQ_i can be set to zero when solving (27b), and the calculated ΔV_i can be ignored. This is highly analogous to the treatment of PV buses in the Z-matrix approach, and suffers from the same weakness. However, where storage is at a premium, and when the method converges, it will be attractive.

VIII. Miscellaneous Load-Flow Methods

This section deals briefly with a selection of proposed methods, not fitting into the popular categories, that have attracted some interest within the last few years.

A. Minimization Methods [40], [41]

In these methods, the load-flow problem is converted into the minimization of an unconstrained scalar objective function f_0, usually equal to the sum of the squares of the current or power mismatches. The global minimum point $f_0 = 0$ coincides with the load-flow solution, since each mismatch must then be zero. The nonlinear programming field provides many minimum-seeking processes from which to choose. The minimization approach does not appear to be well-suited to the basic load-flow calculation, either from a theoretical standpoint or from practical results obtained. Most numerical methods for solving a set of equations are at their most successful when the problem formulation retains as much linearity as possible. In formulation f_0, all the nonlinearities in the mismatch functions become more nonlinear by squaring them.

B. Hybrid Newton/Minimization [42]

This ingenious idea attempts to accelerate Newton's method. After one Newton iteration, a straight line through the first and second iteration points in the multidimensional coordinate space should pass reasonably close to the solution point, if the process is converging. This solution point is also the solution $f_0 = 0$ of the minimization formulation of Section VIII-A based on the sum of the squares of the Newton mismatch functions. A search along the defined line is therefore made for the minimum value of f_0, and once found, this point is accepted as the accelerated Newton iteration point. The scheme can be applied at each Newton iteration. The gradient components of f_0 needed during the search can be obtained from the evaluation of the Newton Jacobian matrix. Nevertheless, the search is time-consuming, and it is probably only worth performing after the first iteration, if at all, since closer to the solution the quadratic Newton process homes in rapidly. However, if the aforementioned straight line fortuitously passes very close to the solution point, the scheme is very successful. The method also guarantees nondivergence, but the interpretation of a located minimum $f_0 \neq 0$ (local or global?) is difficult.

C. Hybrid Newton/Z-Matrix Method [43]

As noted in Section V-C, the attraction of the simultaneous-displacements version of the Z-matrix method is that the sparse triangular factors of the Y matrix (utilizing symmetry) can be used. However, the method's convergence is weak if the system contains PV buses. The hybrid method therefore applies "Z-matrix" iteration only to PQ buses, i.e., removing the rows and columns of voltage-controlled buses

from Y before factorizing it. The polar power-mismatch Newton version is used to solve separately for the voltage angles of PV buses. Block-successive iteration is performed between the Newton and "Z-matrix" iterations. If the number of PV buses is small, the storage required for the matrix solutions is a little more than half of that for a standard Newton solution, and the computing time per iteration is about half as much. On well-conditioned systems, the hybrid method takes about twice as many iterations as Newton's method.

D. *Transient-Response Analog* [44]

If a digital step-by-step transient-response calculation is continued until the system reaches its steady state, this is the solution to the system load-flow problem. This idea has been adapted by expressing each PV-bus active-power constraint as the differential equation

$$d^2\theta_i/dt^2 = \Delta P_i/H_i \tag{28a}$$

where H_i is an artificial "inertia." In one version of the method, PQ buses are eliminated at each step by inserting shunt admittances in the network equivalent to their loads. In another version, each PQ bus is represented by (28a) and the reactive-power constraint

$$d^2V_i/dt^2 = \Delta Q_i/G_i. \tag{28b}$$

The system is solved by step-by-step integration in the usual transient-response manner, where the initial erroneous bus-voltage estimates provide the disturbance to the system. A simple but effective discontinuous damping device is introduced to encourage the dynamic system to settle down to the steady state rapidly.

The authors claim very high reliability of convergence. However, the method is unlikely to be competitive with other modern methods.

IX. Automatic Controls and Adjustments in Load-Flow Solutions

Most practical general-purpose load-flow programs contain adjustment facilities to simulate features of real-life power systems that are not included within the basic problem formulation as in Section III. These features are important, but because they tend, in principle at least, to be simple and do not contribute to the computational elegance of a proposed load-flow algorithm, they are frequently not discussed in the literature. This has tended to obscure their influence on the comparative merits (especially speeds) of the various load-flow methods: the adjustments nearly always increase the number of iterations for a solution, compared with no adjustments.

Many of the features to be represented are automatic power-system devices operating under single-criterion control. The conventional approach to incorporating them in the load-flow solution process is to adjust the relevant parameters and/or variables from iteration to iteration. The precise details of implementing these adjustments can have a major effect on the overall speed of the solution. This section gives some general guides, and describes some of the features most commonly encountered.

A. *Principles of Error-Feedback Adjustment*

We shall consider a single-criterion control in which a parameter x varies in order to maintain a system quantity y at a scheduled value y^{sp}, within some tolerance. The adjustment algorithm takes the form of a closed-loop feedback mechanism, where x is corrected (within limits, if applicable) at successive load-flow iterations, in an attempt to reduce the error $\Delta y = (y^{sp} - y^{cal})$ below the given tolerance. The adjustment process should not be initiated until the load-flow calculation has converged sufficiently to give reliable, though not necessarily very accurate, values for y^{cal}.

The sizes of the corrections Δx must be coordinated carefully with the convergence speed of the particular load-flow method used. If the corrections are too small, overall convergence will be dictated by the slow convergence of the adjustment loop. If the corrections are too big, they will introduce successive perturbations into the load-flow solution that will slow down convergence, or more seriously, produce divergence.

In the physical apparatus, the parameter x is either a continuously variable quantity, or is limited to discrete steps. If in the latter case x is restricted to these discrete values during the adjustments, it is not always easy to avoid the previously mentioned convergence problems. A typical difficulty is hunting between the various adjustments. A common approach is to ignore discreteness in the adjusted parameters until initial overall convergence has been achieved. Only then are the parameters converted to their nearest discrete values, after which the load flow is reconverged with no further adjustments. This can produce minor anomalies, since it may then be found that some of the errors Δy are slightly above their tolerances.

The general "continuous" adjustment formula is simply

$$\Delta x = \alpha \cdot \Delta y \tag{29}$$

where α is the "feedback gain," whose choice is important for each type of control, each load-flow method, and in some cases each system. The objective in choosing α is to minimize the total number of iterations while preserving reliable convergence.

The slowly converging load-flow methods (especially Y-matrix iterative) tend to suffer much less than the rapidly converging ones (Newton and decoupled) from the effects of adjustments. With the former, a small empirical value of α is found, which enables the adjustment algorithm to converge at a similar slow rate to that of the load-flow algorithm. In the rapid load-flow methods, and Newton's in particular, the iteration following a set of adjustments tends to give a fairly accurate load-flow solution. This means that the convergence rates of the various adjustments dictate the overall convergence rate much more than that of the load-flow algorithm. Consequently, the solution takes far more iterations than the corresponding unadjusted solution. For example, a typical adjusted Newton solution will require 10 iterations.

For these rapidly converging load-flow methods, α in (29) should be approximately the sensitivity between x and y at the operating point. For a given system, a suitable fixed estimate of this can be calculated or found empirically. Alternatively, when this is not convenient, the estimation can be built into the adjustment process. When the adjustment process is initiated, a trial correction $\Delta x^{(1)}$ (not too small) is made on the basis of an error $\Delta y^{(0)}$. One or more load-flow iterations are then performed until moderate convergence accuracy is achieved, and the new error is $\Delta y^{(1)}$. An estimate of α can now be calculated thus:

$$\alpha = \Delta x^{(1)}/(\Delta y^{(1)} - \Delta y^{(0)}). \quad (30)$$

Care should be taken that the denominator in (30) is not very small, otherwise α might be too inaccurate. It is useful to impose an upper limit on the change Δx at any one step, in case Δy is initially very large, to avoid perturbing the load-flow algorithm too much.

In some cases the parameter x affects the values of the network admittances. Y-matrix changes can easily be accommodated in load-flow methods that use the admittance elements directly in the network calculation at each iteration, e.g., Y-matrix iterative methods and Newton's method. With other methods, it is more efficient to reflect the x-changes in the nodal injections of the relevant buses.

Since most adjustment processes converge linearly, it is a distinct advantage to choose the tolerance on each Δy to be as large as possible, commensurate with the practical accuracy requirements of the solution.

B. Controlled Transformer In-Phase Taps

An automatic on-load tap-changing transformer usually controls a local bus voltage, within the sensitivity of the voltage relay [6]. Discrete tap-by-tap adjustment is sometimes used, especially in slowly converging load-flow methods where an adjustment is performed only every few iterations. More usually, the continuous approach is used, in which the algorithm for an off-nominal tapping t_i controlling the voltage of bus k becomes

$$\Delta t_i = \alpha_i \cdot \Delta V_k \quad (31)$$

where $\Delta V_k = (V_k^{sp} - V_k^{cal})$. For the rapidly converging load-flow methods, a fixed $\alpha_i = \pm 1$ per-unit is frequently satisfactory, particularly if bus k is a transformer terminal.

Transformer tapping must give precedence to a reactive-power source, if both are designated to control the same bus voltage, i.e., the transformer only taps if the source Q limit is operative. The effect of line-drop compensation can easily be represented by modifying the voltage signal V_k^{cal}. Instead of controlling a bus voltage, the transformer may control the reactive flow through itself or one or more neighboring lines. In this case Q-flow error feedback adjustment is performed.

C. PV-Bus Reactive-Power Limits

When the reactive-power source at a PV bus reaches its upper or lower limit of output and its ability to maintain the scheduled voltage V_i^{sp} is lost, either of two limit-enforcement methods are available: a) the bus is converted to a PQ type with Q_{Gi}^{sp} set to the limit, or b) an error-feedback process of the form (29) is entered, such that at each subsequent iteration, V_i^{sp} is adjusted according to the present limit violation ΔQ_{Gi}:

$$\Delta V_i^{sp} = \alpha_i \cdot \Delta Q_{Gi}. \quad (32)$$

Method a) can be inserted very easily into the Y-matrix iterative and Z-matrix load-flow methods. With Newton methods, bus-type conversion changes the structure of the J-matrix equation, but with most programming schemes the extra work involved is slight, since the equation is constructed and solved at each iteration. In other load-flow methods, bus-type conversion is unattractive where it requires the refactorization of otherwise constant sparse matrices, especially if a number of Q limits are likely to be exceeded at different stages of the calculation. Method b) is then preferred. Both Q-limit methods should provide the opportunity for subsequent limit backoff due to the possible effects of other adjustments.

D. PQ-Bus Voltage Limits

If the voltage magnitude of a PQ bus violates specified limits, reactive-power support may be switched in. This might represent the automatic action of physical apparatus connected to the system. It is also used in planning studies to determine reactive compensation, and is sometimes used as a program device to prevent divergence on difficult systems (the amount of reactive power is gradually reduced as the iterations progress). The adjustments are the reverse of those used for PV-bus Q limits.

E. Automatic Phase-Shift Control

An automatically controlled phase-shifting transformer regulates the flow of active power through itself, or through one or more neighboring lines, by varying its angle of shift ϕ_i. Where the load-flow method takes advantage of matrix symmetry, this shift is best represented by bus-injection techniques. The usual error-feedback adjustment approach is applicable. With the rapidly converging load-flow methods, the scheme (30) has been found to perform very rapidly and reliably. Depending on the type of physical device, the in-phase voltage-ratio variation with ϕ_i should be represented.

F. Area Interchange Control

Each area in a multiarea load-flow problem is likely to have a scheduled net megawatt import/export. This interchange constraint can be satisfied for each area by summing its tie-line flows at each load-flow iteration, and adjusting the scheduled generations. Transfer control is often assigned to one generator bus per area, although it can be shared between any number of generators in the area in some defined proportions. The usual error-feedback adjustment is used. For each area, the change in its generation ΔP_G for an error ΔP_I in the scheduled interchange is

$$\Delta P_G = \alpha \cdot \Delta P_I. \quad (33)$$

Note that $\sum \Delta P_I$ over all areas should be zero. For rapidly converging load-flow methods, $\alpha = 1$ is suitable.

G. Functional Loads

In the basic load-flow formulation, constant PQ-bus loads are normally assumed on the understanding that voltage variation on the PQ bus is compensated for by tap changes at the load side. In their absence, or if the load flow simulates a time period in the system over which the taps do not operate, constant-power loads do not represent the system correctly. In fact, they can sometimes give totally wrong results, e.g., that the system appears to be unstable. One of the main difficulties is in defining the load/voltage characteristics, but where these are available, they are often expressed in the form, for active power,

$$P_{Li}^{sp} = (a + bV_i + cV_i^2)P_{Li(0)}^{sp} \quad (34)$$

and similarly for reactive power.

If the function (34) is substituted for P_{Li}^{sp} in the load-flow equations, it must be adjusted at each iteration as V_i changes. This causes no difficulty—rather, it tends to assist conver-

gence, because in problem terms the function is less nonlinear than a total constant-PQ load of the same nominal size. The term in V^2 can be represented as a fixed shunt admittance. In Newton's method, an addition to the J-matrix diagonal can be made to account for the term in V (note that this is not necessary in formulations using $\Delta P/V$ and $\Delta Q/V$).

H. Automatic Adjustments in Newton's Method

The mathematically general methods such as Newton's and the minimization approach are able to absorb continuous adjustments of the type (29) into the problem formulation, rather than relying on the usual "parasitic" adjustment loops. The adjusted parameters become variables of the main load-flow problem, and are solved for directly by the load-flow algorithm. The possibilities of this approach have been examined in some depth for Newton's method [27], [30]–[32], with the object of benefiting from Newton's powerful convergence while removing the obstacles to obtaining solutions in the "normal" number of iterations. Practical success has been demonstrated with transformers and phase shifters [31], and interarea transfers [30].

Details of the approach cannot be given here. However, for general applications it has two main drawbacks. The first is that it cannot overcome the problem of discontinuities (limits), which inevitably require extra iterations when active. More seriously, the sparsity structure of the Jacobian matrix is usually altered. Controlled devices regulating quantities flowing through themselves or at their terminals do not cause problems, apart from sometimes slightly weakening Newton's convergence properties, but other controls create zero-valued diagonal J-matrix elements. To overcome this, the structure of the J matrix can be altered, at the expense of additional programming complexity, and loss of efficiency in the triangulation process.

X. Multiple-Case Solutions

Many power-system calculations involve the repeated solution of the load-flow problem, with modifications to the network and/or bus constraints in each case. This section gives an introduction to three of the more common applications and techniques, only in so far as the load-flow solutions are concerned.

A. Outage Assessments

Outage assessments are made off-line in system planning and operational planning, and on-line or semi-on-line in operation/control [45] and interactive work [46]. From a given base-case load-flow solution, a sequence of load-flow calculations is carried out, each simulating the loss of one or more transmission elements, generating units or even loads. Since the purpose is to identify potential operating difficulties, high solution accuracy is frequently not required. However, computing speed has a high priority, because of the very large number of cases that may have to be examined. This is especially true in real-time applications, where the reliability of the solutions is also very important. Starting from the base-case solution voltages is nearly always a good approach. The inclusion of adjustments is dependent on the time period after the outage which it is desired to simulate, and the degree of accuracy needed.

The simple dc load flow has been ignored so far in this review, since it is not capable of giving more than a very approximate megawatt flow solution. However, it is found to be adequate in some power systems for assessing megawatt overloads, and has the advantage of being extremely fast. The outage-simulation techniques used in it will also serve as a useful introduction to the similar techniques applied to the ac problem.

The base-case dc load-flow matrix equation, omitting the slack bus, is

$$\theta^B = (B')^{-1} \cdot P^{sp} = D \cdot P^{sp} \tag{35}$$

where the dc admittance matrix B' is the same as that defined for (27a). Outages involving bus powers affect only the vector of bus powers P^{sp}, and are easy to compute. For the outage of a series branch of admittance b between buses i and k, the base-case matrix B' changes. The new matrix is obtained by adding to B' the highly sparse square matrix $b \cdot M^t \cdot M$, where t signifies transpose and M is a null row vector except for $M_i = 1$ and $M_k = -1$ (i or $k \neq s$). Applying the inverse-matrix modification formula, the solution θ^{ik} for this outage case is

$$\theta^{ik} = \theta^B - D^{ik} \cdot M \cdot \theta^B \cdot c = \theta^B - D^{ik} \cdot \theta_{ik}^B \cdot c \tag{36}$$

where D^{ik} is the subtraction of columns i and k of D, and c is the scalar $(1/b + M \cdot D^{ik})^{-1}$. Given the base-case matrix D, the outage solution (36) is performed in little more than n arithmetic operations. The method can easily handle multiple outages.

Matrix D has the usual disadvantages of nonsparsity referred to in Section V-D. An alternative approach is to use the sparse upper triangular factor of B', and use it to generate each vector D^{ik} as required by a repeat solution with M^t as the independent vector. While saving a great deal of storage and computation for D, this approach takes at least three times as long for each outage case (programming to take the sparsity of M^t into account in the solution for D^{ik}). The overall speed comparison between the two cases must be carefully evaluated for a particular application [9], [47]. For approximate ac outage assessments, the same techniques can be applied to the complex equation (20) [48].

For greater accuracy, a true ac load-flow method must be used. The rapidly converging methods such as Newton's and the decoupled versions are generally the most suitable. Since the accuracy requirements of most outage-assessment studies are not too high, however, it is frequently possible to avoid reforming and refactorizing the network matrix or matrices for each branch-outage case, by retaining the matrix factors of the base-case solution throughout. The assumption is made that the outage of a branch changes only its terminal-bus self and mutual elements in the base-case matrix. Compensation for these changes then has to be included in each repeat solution for the unknown voltages using the base-case factors. In the formal Newton method, one approach [49] is to calculate and iterate with the outaged-branch terminal-bus sensitivities, which can be obtained efficiently for all branches at one time [50]. The same technique can be used with decoupled Newton methods.

Those decoupled methods whose network matrices are symmetrical enable the compensation technique as in (36) to be used [37], [38]. For unadjusted moderate-accuracy solutions, such as in security monitoring, a single iteration, started from the base-case voltage solution, is often sufficient, although in general it is desirable to provide the opportunity of reiterating if the accuracy is inadequate (frequently the most

important outage cases). Using the method of (27) for line-overload monitoring, for instance, the following scheme may be used. One outage iteration of (27a) is performed, and on the basis of the new θ's, the branch megawatt flows are obtained. If no lines are approaching overload, this outage case is discontinued; otherwise (27b) is solved, and the solution is iterated to within acceptable mismatches [38].

On the bases of speed, storage, and accuracy in these applications, there seems to be no point in using the formal or decoupled Newton methods in preference to the symmetric decoupled versions. Note that the symmetric decoupled formulations of [35] and [37] give direct solutions for θ and V rather than $\Delta\theta$ and ΔV. These are simply alternative arrangements of the equivalent "Newton-type" equation forms such as that of (27).

B. *Optimal Load Flow* [51]

This section discusses optimal load flow only in the context of methods that use conventional load-flow solutions within the process. An optimal load-flow calculation optimizes the active- and reactive-power dispatch of a system, including as control variables those single-criterion-control parameters that are adjusted during an ordinary load-flow solution. All the relevant static limit constraints on the system operation are enforced.

Some practical approaches to this problem, including the now classical Dommel and Tinney method [52], adjust the control variables in some optimum-seeking manner in between conventional load-flow solutions. Polar-coordinate load-flow formulations are the most natural choice for the optimization application, since voltage magnitudes in particular are then explicitly available as variables for control and limit-enforcement purposes.

Apart from the effects of transmission-apparatus parameter adjustments, the network remains fixed in admittance values and configuration (although outage constraints can also be incorporated [53]). Each resolution of the load flow to calculate the effects of the control-variable adjustments uses the previous point to start it. If the adjustments are not large, a rapidly converging method such as Newton's will give high accuracy in one or two iterations. Newton's method is also valuable because its Jacobian matrix contains the information needed to determine the next incremental adjustments of the control variables [52]. If the latest J matrix from the load-flow solution is to be used, its lower triangular factor must be stored. The fast decoupled method of Section VII-C has also been used for the load-flow solutions and in calculating the incremental costs (Lagrange multipliers) [54]. It requires less storage, and, being much faster than Newton's method for practical accuracies, it gives an overall speed advantage in the optimal load flow of more than 2:1.

C. *Multimachine Dynamic Response Calculations*

The conventional approach to the calculation of power-system stability uses a step-by-step process in which the machine differential equations are integrated alternately with a load-flow solution of the transmission system. The solution methods and degrees of system representation vary widely, and it is only possible here to make some general comments on the load-flow portion of the overall calculation [55].

The load-flow problem to be solved after or during each integration step is characterized by multiple slack buses. Each of these represents the temporarily fixed complex voltage behind the relevant machine impedance(s), transformed into the network reference frame. The convergence powers of all load-flow methods are enhanced by having many slack buses, which are easy to handle in the respective load-flow programs. The load-flow solutions are also assisted by representing loads or parts of loads by fixed shunt admittances, by network reduction (where appropriate), and by the good starting voltages available from the last solution (except after switching operations).

Saliency in the machine impedances introduces time-varying nonbilateral network branches, usually Norton shunts [56]. The variations of these impedances can be represented by bus-injection techniques if the load-flow method relies on a constant and/or symmetrical network matrix.

Often, some or all bus power constraints are represented as fixed shunt admittances. Network reduction can then be undertaken, but the degradation of possible network-sparsity exploitation must be weighed carefully against the expected improvement in load-flow convergence. Nonadmittance loads must be corrected at each iteration (see Section IX-G), and can give convergence problems, particularly constant-power loads, if their bus voltages go very low or high.

Finally, the integration step sizes used can affect the choice of load-flow method. Small step sizes, where the variation of system conditions over the step is small, do not impose such stringent requirements on the convergence power of the load-flow method.

The Y-matrix iterative methods can very conveniently handle the whole range of system representational requirements, and need minimum storage, depending on whether any form of network reduction is performed. They are best suited to small integration step lengths. The method of Section IV-G seems to be advantageous for its reliability, since there are no PV buses. Z-matrix methods have also been widely used, and profit from the same fact. The sparse-factor form will usually be preferable to the explicit inverse, although the latter's ability to iterate with successive displacements gives it an advantage for difficult cases.

Newton's method is suitable for larger integration step lengths, and can handle saliency (the J matrix is nonsymmetrical and is constructed repeatedly) and nonadmittance loads well. The rectangular current-mismatch version is probably best [56], since there are no PV buses, and loads represented as shunt admittances present it with a no-load situation (see Section VI-D). Under these conditions, the method is the same as (22), apart from the time-varying machine-impedance terms in it. If the J matrix is formed and triangulated at each iteration, one iteration per solution is usually sufficient, depending on the step size and integration approach. However, this is still computationally expensive, and constant J-matrix upper and lower triangular factors, containing mean values of the machine impedances, can be used over many integration steps without updating.

It should be noted that for large-disturbance studies, the decoupled methods are not appropriate, since the decoupling principle will be violated.

XI. Concluding Remarks

This review has tried to make the necessary compromise between presenting a very large number of the available load-flow techniques, and giving due attention to the relevant principles and computational implications. For the most part, therefore, it has been possible to consider only the mainstream

ideas and algorithms. Scores of other methods and variants, some of them highly original, have had to be omitted. In addition, a number of other specific aspects of the load-flow problem have not been touched on.

One of the topics not covered is that of system tearing. This subject lost a lot of impetus in the load-flow application after the introduction of sparse matrix methods (with which it is not incompatible, however). The sizes of problems to be solved have increased at a faster rate than the development of cheaper computer core memory elements. With this, and the prospects of multicomputer applications in power systems, diakoptical [57] and block-iteration [58] techniques for load flow have been receiving fresh attention. Tearing is also one way of trying to avoid excessive fillup in the sparse-matrix triangulation process. The subject of load-flow equivalents is currently of great importance, especially for real-time control and monitoring. Another topic not mentioned in the review is the efficient simulation of controlled HV dc links in the load-flow calculation. The effect of HV dc links on the ac-system calculation is similar to the adjustment problem described in Section IX. A more fundamental issue is the nonuniqueness of load-flow solutions [59]. This is when two or more different solutions, each equally feasible from an engineering viewpoint, can be obtained. Although, fortunately, this phenomenon arises infrequently in practice, there seems to be no reliable and inexpensive way of identifying cases where several such solutions exist, nor is there a criterion for determining which solution most properly represents the likely operation of the system. Finally, the question of data uncertainty and its effects on the credibility of the calculated load flows is important, and is currently being examined [60]–[62].

Algorithmic improvements in load-flow calculation over the years have tended to be measured by speed, rather more than by the other criteria of goodness listed in Table I. For the traditional off-line single-case solution, algorithmic speed has ceased to be relevant. Using well-programmed versions of the modern methods with sparse-matrix triangulation, input /output is much more the limiting time factor than the iterative solution process, and should therefore be given more attention in the design and coding of programs (NB, the fastest algorithms and computers can now perform accurate iterative solutions for 1000-bus load flows in under one CPU-second). The need for speed remains, however, for repeated solutions such as in outage-security assessment, especially in real-time applications, and where the dc load flow does not give an acceptable minimum accuracy.

Storage is a perennial problem for many users, for two main reasons. Small computers are now fast enough to solve medium–large systems. Those organizations with large computers tend to want to solve very large systems, up to several thousand buses. Moreover, computer managers are usually reluctant to allocate most or all of the core area to a single (and noncommercial) job. This problem might be combatted by developing even more effective ordering methods in sparsity exploitation (for very large systems), and/or system tearing applications. Unless speed is of the essence, a limited number of block data transfers to and from fast backing store may be tolerable. Of the modern algorithms, the formal Newton method is the most storage-consuming, especially when the lower triangular factor of the Jacobian matrix is needed. The efficient row-elimination approach to triangulation is not well-suited to block transferring. The fast decoupled method offers some scope, since with about 25 transfers per iterative solution, its storage requirements are similar to those of an in-core Y-matrix iterative solution.

Some of the modern load-flow algorithms converge comparatively very reliably, but no known method that is fast enough for routine application can guarantee to solve any feasible problem. One of the main values of absolute reliability is that, in the event of failure to converge, the engineer can then assume with confidence that the data or the system modeled are inadequate. For operational purposes, it is probably enough that the method should not break down for any physically operable system condition. Newton's method, with enhancements, and the fast decoupled method seem to be approaching this requirement. For planning, even greater reliability properties are needed, since the engineer will sometimes want to solve cases that do not represent practical operation, e.g., to investigate stability margins.

The Newton methods are certainly the most versatile methods available. They have been used in a wide variety of system optimization calculations, they provide sensitivity analyses, and they can be used in modern dynamic-response and outage-assessment calculations. As far as adjustments are concerned, the case for pursuing the formal incorporation of single-criterion controls into Newton's method does not appear to be strong, with the present state of art in sparse matrix techniques. Only a restricted set of controls can be thus included efficiently, and without considerable extra complications. As soon as one conventional adjustment has to be introduced into a program, the benefits of the formally incorporated controls are largely lost. A better approach might be to retain the simple conventional adjustment philosophy throughout, with careful implementation. Where speed is still important, the fast decoupled load-flow method can be used. An adjusted solution that takes say 12 iterations with this method is equivalent in time to 3 iterations of the standard Newton method.

This review is concluded by emphasizing the importance of sparse-matrix techniques in the exploitation and further development of load-flow algorithms. Modern methods using sparse-matrix factorization need great attention to programming efficiency if they are to fully realize their considerable advantages over previous methods. The onus is therefore placed on the load-flow analyst to gain sufficient understanding of the details and implications of these techniques.

References

General

[1] M. R. G. Al-Shakarchi, "Nodal iterative load flow," M.Sc. thesis, University of Manchester Institute of Science and Technology, Manchester, U. K., 1973. A copy of the bibliography on load-flow calculation contained in this dissertation is available from B. Stott.
[2] D. A. Conner, "Representative bibliography on load-flow analysis and related topics," presented at the IEEE PES Winter Meet., New York, Jan. 28–Feb. 2, 1973. Paper C73-104-7.
[3] W. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proc. IEEE*, vol. 55, pp. 1801–1809, Nov. 1967.
[4] M. A. Laughton and M. W. Humphrey Davies, "Numerical techniques in the solution of power system load flow problems," *Proc. Inst. Elec. Eng.* vol., 111, pp. 1575–1588, Sept. 1964.
[5] A. M. Sasson and F. J. Jaimes, "Digital methods applied to power flow studies," *IEEE Trans. Power App. Syst.*, vol. PAS-86, pp. 860–867, July 1967.
[6] G. W. Stagg and A. H. El-Abiad, *Computer Methods in Power System Analysis*. New York: McGraw-Hill, 1968.
[7] O. Elgerd, *Electric Energy Systems Theory*. New York: McGraw-Hill, 1971.
[8] A. Ralston, *A First Course in Numerical Analysis*. New York: McGraw-Hill, 1965.
[9] W. F. Tinney and W. L. Powell, "Comparison of matrix inversion

and sparse triangular factorization for solution of power network problems," presented at Romania/U.S.A. Research Seminar, Bucharest, Romania, June 3–6, 1974.

Y-Matrix Iterative Load-Flow Methods

[10] J. B. Ward and H. W. Hale, "Digital computer solution of power-flow problems," *AIEE Trans. (Power App. Syst.)*, vol. 75, pp. 398–404, June 1956.

[11] A. F. Glimn and G. W. Stagg, "Automatic calculation of load flows," *AIEE Trans. (Power App. Syst.)*, vol. 76, pp. 817–828, Oct. 1957.

[12] R. J. Brown and W. F. Tinney, "Digital solutions for large power networks," *AIEE Trans. (Power App. Syst.)*, vol. 76, pp. 347–355, June 1957.

[13] D. G. Taylor and J. A. Treece, "Load flow analysis by the Gauss-Seidel method," presented at the Symp. on Power Systems Load Flow Analysis, University of Manchester Institute of Science and Technology, Manchester, U. K., 1967.

[14] C. Trevino, "Cases of difficult convergence in load flow problems," presented at IEEE Winter Power Meet., New York, Jan. 31–Feb. 5, 1971. Paper 71 CP 62-PWR.

[15] F. J. Hubert and D. R. Hayes, "A rapid digital computer solution for power system network load-flow," *IEEE Trans. Power App. Syst.*, vol. PAS-90, pp. 934–940, May/June 1971.

[16] W. W. Maslin, S. T. Matraszek, C. H. Rush, and J. G. Irwin, "A power system planning computer program package emphasizing flexibility and compatibility," presented at the IEEE Summer Power Meet., Los Angeles, July 1970. Paper 70 CP 684-PWR.

[17] R. Podmore and J. M. Undrill, "Modified nodal iterative load flow algorithm to handle series capacitive branches," *IEEE Trans. Power App. Syst.*, vol. PAS-92, pp. 1379–1387, July/Aug. 1973.

[18] J. A. Treece, "Bootstrap Gauss-Seidel load flow," *Proc. Inst. Elec. Eng.*, vol. 116, pp. 866–870, May 1969.

Z-Matrix Load-Flow Methods

[19] P. P. Gupta and M. W. Humphrey Davies, "Digital computers in power system analysis," *Proc. Inst. Elec. Eng.*, vol. 108A, pp. 383–404, Jan. 1961.

[20] A. Brameller and J. K. Denmead, "Some improved methods of digital network analysis," *Proc. Inst. Elec. Eng.*, vol. 109A, pp. 109–116, Feb. 1962.

[21] H. E. Brown, G. K. Carter, H. H. Happ, and C. E. Person, "Power flow solution by impedance matrix iterative method," *IEEE Trans. Power App. Syst.*, vol. PAS-82, pp. 1–10, Apr. 1963.

[22] ——, "Z-matrix algorithms in load-flow programs," *IEEE Trans. Power App. Syst.*, vol. PAS-87, pp. 807–814, Mar. 1968.

[23] L. L. Freris and A. M. Sasson, "Investigations on the load-flow problem," *Proc. Inst. Elec. Eng.*, vol. 114, p. 1960, Dec. 1967.

Newton Load-Flow Methods

[24] J. E. Van Ness, "Iteration methods for digital load flow studies," *AIEE Trans. (Power App. Syst.)*, vol. 78, pp. 583–588, Aug. 1959.

[25] J. E. Van Ness and J. H. Griffin, "Elimination methods for load-flow studies," *AIEE Trans. (Power App. Syst.)*, vol. 80, pp. 299–304, June 1961.

[26] W. F. Tinney and C. E. Hart, "Power flow solution by Newton's method," *IEEE Trans. Power App. Syst.*, vol. PAS-86, pp. 1449–1456, Nov. 1967.

[27] H. W. Dommel, W. F. Tinney, and W. L. Powell, "Further developments in Newton's method for power system applications," presented at IEEE Winter Power Meet., New York, Jan. 25–30, 1970. Paper 70 CP 161-PWR.

[28] H. W. Dommel, discussion of [26], *IEEE Trans. Power App. Syst.*, vol. PAS-86, pp. 1456–1458, Nov. 1967.

[29] B. Stott, "Effective starting process for Newton-Raphson load flows," *Proc. Inst. Elec. Eng.*, vol. 118, pp. 983–987, Aug. 1971.

[30] J. P. Britton, "Improved area interchange control for Newton's method load flows," *IEEE Trans. Power App. Syst.*, vol. PAS-88, pp. 1577–1581, Oct. 1969.

[31] N. M. Peterson and W. S. Meyer, "Automatic adjustment of transformer and phase-shifter taps in the Newton power flow," *IEEE Trans. Power App. Syst.*, vol. PAS-90, pp. 103–108, Jan./Feb. 1971.

[32] J. P. Britton, "Improved load flow performance through a more general equation form," *IEEE Trans. Power App. Syst.*, vol. PAS-90, pp. 109–116, Jan./Feb. 1971.

Decoupled Load-Flow Methods

[33] G. W. Stagg and A. G. Phadke, "Real-time evaluation of power-system contingencies—detection of steady state overloads," presented at the IEEE Summer Power Meet., Los Angeles, July 1970. Paper 70 CP 692-PWR.

[34] B. Stott, "Decoupled Newton load flow," *IEEE Trans. Power App. Syst.*, vol. PAS-91, pp. 1955–1959, Sept./Oct. 1972.

[35] S. T. Despotovic, B. S. Babic, and V. P. Mastilovic, "A rapid and reliable method for solving load flow problems," *IEEE Trans. Power App. Syst.*, vol. PAS-90, pp. 123–130, Jan./Feb. 1971.

[36] K. Uemura, "Power flow solution by a Z-matrix type method and its application to contingency evaluation," presented at IEEE PICA Conf., Boston, 1971.

[37] N. M. Peterson, W. F. Tinney, and D. W. Bree, Jr., "Iterative linear ac power flow solution for fast approximate outage studies," *IEEE Trans. Power App. Syst.*, vol. PAS-91, pp. 2048–2056, Sept./Oct. 1972.

[38] B. Stott and O. Alsac, "Fast decoupled load flow," presented at IEEE PES Summer Meet., Vancouver, July 15–20, 1973. Paper T 73 463-7. Also, *IEEE Trans. Power App. Syst.*, vol. PAS-93, pp. 859–869, May/June 1974.

[39] E. Hobson, discussion of [38], *IEEE Trans. Power App. Syst.*, vol. PAS-93, p. 868, May/June 1974.

Miscellaneous Load-Flow Methods

[40] Y. Wallach, "Gradient methods for load flow-problems," *IEEE Trans. Power App. Syst.*, vol. PAS-87, pp. 1314–1318, May 1968.

[41] A. M. Sasson, "Nonlinear programming solutions for the load-flow, minimum-loss, and economic dispatching problems," *IEEE Trans. Power App. Syst.*, vol. PAS-88, pp. 399–409, Apr. 1969.

[42] A. M. Sasson, C. Trevino, and F. Aboytes, "Improved Newton's load flow through a minimization technique," presented at IEEE PICA Conf., Boston, 1971.

[43] Y. P. Dusonchet, S. N. Talukdar, H. E. Sinnott, and A. H. El-Abiad, "Load flows using a combination of point Jacobi and Newton's methods," *IEEE Trans. Power App. Syst.*, vol. PAS-90, pp. 941–949, May/June 1971.

[44] R. H. Galloway, J. Taylor, W. D. Hogg, and M. Scott, "New approach to power-system load-flow analysis in a digital computer," *Proc. Inst. Elec. Eng.*, vol. 117, pp. 165–169, Jan. 1970.

Multiple-Case Load-Flow Solutions

[45] T. E. Dy Liacco, "Real-time computer control of power systems," this issue, pp. 884–891.

[46] J. M. Undrill, F. P. deMello, T. E. Kostyniak, and R. J. Mills, "Interactive computation in power system analysis," this issue, pp. 1009–1018.

[47] B. Fox and A. M. Revington, "Network calculations for on-line control of a power system," in *Proc. IEE Conf. on Computers in Power System Operation and Control* (Bournemouth, U. K.), pp. 261–275, 1972.

[48] H. E. Brown, "Contingencies evaluated by a Z-matrix method," *IEEE Trans. Power App. Syst.*, vol. PAS-88, pp. 409–412, Apr. 1969.

[49] M. S. Sachdev and S. A. Ibrahim, "A fast approximate technique for outage studies in power system planning and operation," presented at IEEE PES Summer Meet., Vancouver, July 15–20, 1973. Paper T 73 469-4.

[50] K. Takahashi, J. Fagan, and M. Chen, "Formation of a sparse bus impedance matrix and its application to short circuit study," presented at IEEE PICA Conf., Minneapolis, 1973.

[51] A. M. Sasson and H. M. Merrill, "Power system optimization techniques," this issue, pp. 959–972.

[52] H. W. Dommel and W. F. Tinney, "Optimal power flow solutions," *IEEE Trans. Power App. Syst.*, vol. PAS-87, pp. 1866–1876, Oct. 1968.

[53] O. Alsac and B. Stott, "Optimal load flow with steady-state security," presented at IEEE PES Summer Meet., Vancouver, July 15–20, 1973. Paper T 73 484–3.

[54] B. Stott and O. Alsac, closure of [38], *IEEE Trans. Power App. Syst.*, vol. PAS-93, p. 869, May/June 1974.

[55] B. Stott, "Formulation and solution of matrix equations in multimachine stability calculations," presented at Symp. on Power System Dynamics, University of Manchester Institute of Science and Technology, Manchester, U. K., Sept. 1973.

[56] H. W. Dommel and N. Sato, "Fast transient stability solutions," *IEEE Trans. Power App. Syst.*, vol. PAS-91, pp. 1643–1650, July/Aug. 1972.

Further Aspects of Load Flow

[57] H. H. Happ, "Diakoptics—The solution of system problems by tearing," this issue, pp. 930–940.

[58] W. E. Bosarge, J. A. Jordan, and W. A. Murray, "A non-linear block SOR-Newton load flow algorithm," presented at IEEE PES Summer Meet., Vancouver, July 15–20, 1973. Paper C 73 644-5.

[59] A. J. Korsak, "On the question of uniqueness of stable load-flow solutions," *IEEE Trans. Power App. Syst.*, vol. PAS-91, pp. 1093–1100, May/June 1972.

[60] L. S. Van Slyck and J. F. Dopazo, "Conventional load flow not suited for real-time power system monitoring," presented at IEEE PICA Conf., Minneapolis, Minn., 1973.

[61] B. Borkowska, "Probabilistic load flow," presented at IEEE PES Summer Meet., Vancouver, July 15–20, 1973. Paper T 73 485-0.

[62] J. F. Dopazo, O. A. Klitin and A. M. Sasson, "Stochastic load flows," presented at IEEE PES Summer Meet., Anaheim, Calif., July 14–19, 1974. Paper T 74 308–3.